

A CLASSIFICATION OF TREES AND
APPLICATIONS OF TOPOLOGY AND GRAPH THEORY TO
NEUROSURGERY

By

TAE-IL YI

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2000

Copyright 2000

by

TAE-IL YI

TO

My parents, **Beom-Young & Bong-Wol**,

who show me how to love,

My wife, **Moon-Sil**,

who shows me what is love,

and

My children, **David & Peter**,

who are the reasons why I love

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to the members of my supervisory committee, Dr. Beverly Brechner, Dr. Philip Boyland, Dr. James Keesling, Dr. Li-Chien Shen, and Dr. Mary Kantowski, for their patience, knowledge, concern, and especially for showing me how to study and teach. I would especially like to thank my advisor, Dr. Beverly Brechner, who accepted me and gave me a chance to work with her. I also want to express my appreciation for all her effort and all that she has done for me, not only in my major field, but also in my life. I would like to thank Matthew Harvey who brought me some wonderful ideas and wrote the *Mathematica* programs. I would like to thank Dr. Francis Bova who brought me these questions which were 'new world' to me. I would like to thank Dr. Meeks for helpful conversations, and Dr. Thomas Wagner who gave me the information about neurosurgery that I did not know before. I would like to thank Dr. Yunmei Chen who gave me inspiration. I would like to thank Dr. Miklos Bona who showed me some insights of combinatorics. Finally, I would like to thank the Department of Mathematics that allowed me to be here and meet all these wonderful people.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
ABSTRACT	vii
CHAPTERS	
1 INTRODUCTION	1
2 AN AUTOMATED SPHERE PACKING PLAN FOR BRAIN TUMORS	5
2.1 Introduction	5
2.2 Linac	5
2.3 Procedure of Radiosurgery	6
2.4 Making a Sphere by Arcs of Beams	10
2.5 Sphere Packing Plan	10
2.6 Shelling Algorithm	15
3 A CLASSIFICATION OF UNLABELED TREES	25
3.1 Introduction	25
3.2 Definitions	26
3.3 Vertex Sequence	28
3.4 Degree Sequence	36
3.5 Perfect Sequence and Main Theorem	44
3.6 Obtaining Perfect Sequences	45
3.7 Extended Partitions and the Algorithm	53
4 TREES AS INVARIANTS FOR BRAIN TUMOR SHAPES	59
4.1 Introduction	59
4.2 Definitions and Some Properties	60
4.3 Graphs for Sphere Packing Plans	64
4.4 Order in the Vertex Set	65
4.5 Maximal Tree from a Graph	71
4.6 Graph from a Maximal Tree	81
4.7 Algorithm for Our Maximal Tree	81

APPENDICES

A	SHELLING PROGRAM FOR SPHERE PACKING	82
A.1	Program for Data Reading	82
A.2	Program for Center Searching	83
B	PROGRAM FOR THE MAXIMAL TREE OF A GRAPH	91
C	PROGRAM FOR CLASSIFYING TREES BY SEQUENCES	93
C.1	Program	93
C.2	Some Program Results	97
D	LABELED GRAPHS AND TREES	102
D.1	Number of Labeled Graphs and Labeled Trees	102
D.2	Sketch of the Construction of a Prüfer-code	102
D.3	Reconstruction of a Tree from a Prüfer-code	103
REFERENCES		105
BIOGRAPHICAL SKETCH		105

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

A CLASSIFICATION OF TREES AND
APPLICATIONS OF TOPOLOGY AND GRAPH THEORY TO
NEUROSURGERY

By

Tae-II Yi

August 2000

Chairman: Dr. Beverly L. Brechner
Major Department: Mathematics

The work of this dissertation was motivated by our work with a medical group at the Brain Institute at the University of Florida. This group was led by Dr. Francis Bova, who raised the following questions:

(1) Can you automate the time-consuming sphere packing process for our “multiple isocenters method of stereographic radiosurgery” for brain tumors? (This surgery is noninvasive, radiation surgery to destroy brain tumors.)

(2) Can you classify the shapes of brain tumors?

In this dissertation, we construct some algorithms and programs to (partially) answer these questions. Our major (joint) results are described below.

Our first result is an algorithm and a program that produce a unique sphere packing plan for any given brain tumor shape, filling the shape with different size spheres in a unique way. This process automates the ‘sphere packing’ part of the

surgery plan. Additional code by the medical group was necessary to allow for physician input in certain critical cases. It was then possible for the medical group to incorporate this into their surgery plan, semi-automating it, and significantly reducing the surgeons' surgery planning time.

Our second result is a complete classification of unlabeled n -trees. We assign to each n -tree, a sequence of n nonnegative integers satisfying certain properties; and from each such sequence, we can recover the n -tree assigned to it. In fact, we present both an algorithm and code to produce not only the sequences, but also pictures of the corresponding trees.

Our final result is a first attempt at a topological classification of tumor shapes. We first use our sphere packing plan to construct a graph representing the tumor. We then have an algorithm and program which uniquely select a maximal tree in the graph. Thus we can assign a unique tree to each tumor shape. However, different graphs may produce the same tree. Our tree classification allows us to compare two tumor shapes very easily, by just looking at their corresponding sequences. In particular, different sequences come from different sphere packing plans, but the converse does not hold.

CHAPTER 1 INTRODUCTION

The work of this dissertation was motivated by two questions raised by Dr. Francis Bova, of the Brain Institute, Shands Hospital, and the Department of Neurosurgery, all at the University of Florida. It turns out the different-sounding questions are very much related and are tied together naturally in this thesis. In fact, our results for the second question depend upon our results for the first question. Dr. Bova led a medical group, which joined with our mathematics group, for a number of discussions over a two year period. This resulted, for us, in both a joint medical paper [11] and in this dissertation. We also want to reference the dissertation [10] of Thomas Wagner, Dr. Bova's medical physics student, who worked with us.

We expect to have three mathematics publications from this dissertation—from Chapters 2, 3, and 4, respectively. The paper resulting from Chapter 2 will be joint with Professor Beverly Brechner, Professor Yunmei Chen, and Dr. Thomas Wagner. The papers resulting from Chapters 3 and 4 will be joint with Matthew Harvey, an undergraduate student working with Professor James Keesling on his honors thesis. Further, we anticipate an additional paper, to be joint with Yunmei Chen and her student, Dr. Stacey Chastain. This anticipated work will be an analysis of the tumor shapes of approximately 1000 interesting tumor radiation surgeries done by Bova and his colleagues at the University of Florida.

Dr. Bova's questions were the following:

1. Is it possible to automate the medical group's "multiple isocenters method of stereographic radiosurgery" to destroy brain tumors?
2. Is it possible to classify brain tumor shapes?

In this dissertation, we develop some algorithms and programs related to these questions. In stereographic radiosurgery, which is noninvasive surgery, radiation is delivered by means of collimators—cylindrical metal tubes, with cylindrical holes through them, through which the radiation flows. The goal of stereotactic radiosurgery for a brain tumor is to deliver the desired dosage to the target, and only the target. This is not possible in reality. So they do the next best thing, which is to deliver enough dosage to the target, to avoid as much normal tissue as possible, and to deliver as little radiation as possible to whatever normal tissue must be affected. There are two additional important criteria—dose homogeneity and dose conformality. That is, we do not want ‘hot spots,’ which have been experimentally determined to cause complications; and we do want rapid falloff of dose levels outside the actual tumor. One of several such radiation surgery methods is called the ‘Multiple Isocenter Method.’ This involves filling the tumor image with spheres of different sizes, until the image is best filled up. In this dissertation, we call such a plan, a ‘Sphere Packing Treatment Plan.’

In Chapter 2, we develop both an algorithm, which is called a ‘shelling process’ (or ‘grassfire process’ in medical terminology), and a C++ program to identify the centers and sizes of spheres for this sphere packing treatment plan. For any given brain tumor, our program gives us a unique sphere packing plan. This sphere packing plan can be represented numerically as a list of parameters of the spheres, namely the centers and radii of the spheres. Based on this algorithm, the medical group developed a proprietary program for an automated sphere packing treatment plan, including dosage distribution, and other things (see Wagner et al.[11]). Since this program always produces the same plan for the same tumor, it can be used as a ‘benchmark’ against which to compare other treatment plans.

In Chapter 3, we obtain a code to denote an unlabeled tree by a unique, ordered, nonnegative integer sequence satisfying certain properties. We call such

sequences, *perfect sequences*. That is, we obtain a complete classification of unlabeled n -trees by means of the perfect sequences of n nonnegative integers. Thus we have our main theorem of this chapter, the Tree Classification Theorem, (Theorem 3.5.5): *For any positive integer n , let $\mathcal{T}(n)$ be the set of unlabeled n -trees and $\mathcal{P}(n)$ the set of perfect sequences of n -trees. Then there is a one-to-one correspondence between $\mathcal{T}(n)$ and $\mathcal{P}(n)$.* We show how to assign a perfect sequence to a given tree, and conversely, how to reconstruct a tree from a given perfect sequence. We also include an algorithm and a *Mathematica* program to produce not only all the perfect n -sequences, for any given positive integer $n \geq 3$, but also pictures of the corresponding trees. By means of this program, we get our classification of unlabeled n -trees and also the total number of n -trees.

In Chapter 4, we make a first attempt at a topological shape classification of tumors—actually of the sphere packing plan associated with the tumor shape. We first assign to each tumor a unique graph, and from that, we determine a unique tree. We obtain the graph by assigning a vertex for each sphere, and an edge between each pair of vertices representing adjacent spheres. This produces a unique connected graph corresponding to each sphere packing plan. Then, by using the notion of *cutvertex*, we give an order to the vertex set. We then use this order to build a unique *maximal tree* contained in the graph. Next, we develop an algorithm and a *Mathematica* program to assign a unique tree to each sphere packing plan. This program produces a list of edges of the tree, making it clear which edges are incident.

Using our results of Chapter 3, we can now assign exactly one perfect sequence to each tumor. However, it may happen that more than one tumor shape (or graph) gives rise to the same tree, and therefore to the same perfect sequence. Thus, we do not yet have a real classification of tumor shapes. However, if two perfect sequences are distinct, then their corresponding trees, and therefore their respective corresponding graphs and sphere packing plans, are also distinct. Therefore we may consider

their shapes to be distinct. Thus, we have our main result of Chapter 4, Theorem 4.5.6: *The perfect sequences are invariants of the shapes of arbitrary brain tumors.* That is, if two brain tumors are represented by distinct perfect sequences, then their corresponding trees are not isomorphic, and therefore their respective graphs and sphere packing plans are not isomorphic. Thus, we may consider their shapes to be distinct.

We refer the reader to Douglas[1], Gross[3], and Harary[5] for graph theory. We also refer the reader to Hall[4], Riordan[7] and Stanley[9] for combinatorics.

CHAPTER 2 AN AUTOMATED SPHERE PACKING PLAN FOR BRAIN TUMORS

2.1 Introduction

The goal of stereotactic radiosurgery for a brain tumor is to deliver the desired dosage to the target, and only the target. This is not possible in reality. So they do the next best thing, which is to deliver enough dosage to the target, to avoid as much normal tissue as possible, and to deliver as little radiation as possible to whatever normal tissue must be affected. There are two additional important criteria—dose homogeneity and dose conformity. That is, we do not want ‘hot spots,’ which have been experimentally determined to cause complications; and we do want rapid falloff of dose levels outside the actual tumor. One of several such radiation surgery methods is called the ‘Multiple Isocenter Method.’ This involves filling the tumor image with spheres of different sizes, until the image is best filled up. This noninvasive method of surgery, namely by using radiation, relies on a piece of equipment called the *Linear Accelerator* (or simply, *Linac*). In this chapter, we show how to automate the construction of a sphere packing plan for the tumor image. Most of the information in this chapter about treatment of a brain tumor is taken from Friedman et al.[2]. See [2] for further reference.

2.2 Linac

According to Friedman et al.[2], the linear accelerator is a complex machine capable of producing X-rays. A large amount of energy is generated by the power supply, which then powers the filament shown. This causes electrons to be emitted by the filament, which are in turn accelerated to higher energies using a (microwave)

wave guide. The electrons are then changed in direction by the magnet so that they impact on a heavy metal alloy target. This results in X-ray production that can then be collimated or shaped by both primary and secondary collimators within the linear accelerator head. This beam is further collimated for radiosurgery by the tertiary radiosurgery collimator. (See Figure 2-1 and Figure 2-2.)

The Linac is mounted on a rotating gantry such that the beam has a center of rotation about 1.5m above the floor. Usually, the isocenter accuracy is defined within a 2mm sphere. Because stereotactic radiosurgery depends on optimized accuracy, an improved system was designed at the University of Florida, by adding a set of bearings to the stereotactic collimator system and under the patient table. As a result, this new system achieves mechanical accuracy within $0.2\text{mm} \pm 0.1\text{mm}$ for defining the treatment isocenter of beam delivery.

The tertiary collimators are generally circular and allow improved centering of the treatment beam. The sizes of these collimators are from 5 to 40mm in 2- to 5-mm increments.

2.3 Procedure of Radiosurgery

The procedure of radiosurgery is separated into 5 steps as follows.

- (1) Head ring attachment
- (2) CT scan
- (3) Image Processing
- (4) Planning
- (5) Treatment delivery

For more detailed information, refer to Friedman et al.[2].

- (1) Head ring attachment

The stereotactic radiosurgery treatment requires attachment of a stereotactic head ring. This ring provides accurate information about the position and the shape

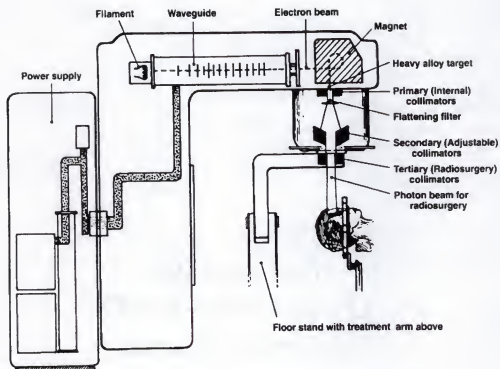


Figure 2-1. System of Linac (Friedman et al.[2])

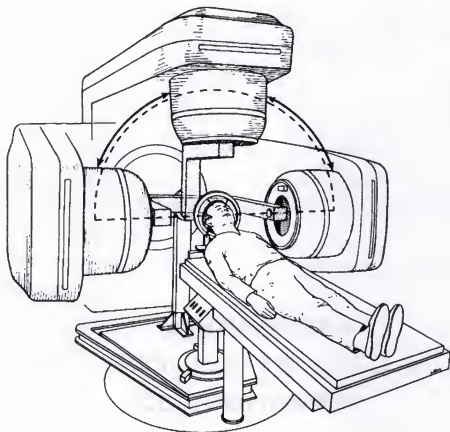


Figure 2-2. Operation of Linac (Friedman et al.[2])

of brain tumors from the computer tomography (CT). This ring also helps to insure that the treatment target is accurately placed at the precise isocenter of the Linac.

(2) CT scan

After attaching the head ring, a series of cut filming is taken. Eighty to 150 CT slices are taken per patient. For radiosurgical treatment planning, getting the accurate size and shape of the target is very important. Underestimation of the target size may result in treatment failure. Overestimation of size results in the inclusion of normal brain tissue within the treatment volume. Misrepresentation of an irregular target shape may lead to radiation damage of normal brain tissue. The accuracy of CT scanning depends on the size of the pixels that make up the scan image. The lateral and AP accuracy of scanning is approximately 0.6mm. But the axial dimension is determined by slice thickness of the CT scan, which is 1mm in general. The physicians use the so-called Brown-Roberts-Wells coordinates—or, simply BRW coordinates—which are called lateral (Lat), anterior-posterior (AP), and axial (Ax) coordinates. These are the x, y, and z coordinates, respectively. The direction of the lateral coordinate is from the left hand side ear to the right hand side ear of the patient. So the lateral view (or *sagittal*) is similar to the side view of the head, that is, views of slices along the lateral axis. This is the same as yz-plane views. The direction of the AP coordinate is from the back side of the head to the face. So the AP view (or *coronal*) is similar with the front view of the head, that is, views of slices along the AP axis. This is the same as xz-plane views. The direction of the axial (or vertical) coordinate is from the neck to the top of the head. So the axial view (or *trans-axial*) is similar to the sky view of the head, that is, views of slices along the axial axis. This is the same as xy-plane views. (See Figure 2-4.)

(3) Image Processing

An MRI scan has been done the day before surgery. Using a mouselike device, the CT and MRI images are transferred to the computer screen and fused. Then,

using geometric equations, the computer determines the lateral, AP, and vertical stereotactic coordinates of each point in each CT slice.

(4) Planning

Physicians use a mouse and their own judgment to 'eyeball' the location and the size of spheres for the treatment plan.

(5) Treatment delivery

The geometric sphere packing plan, developed as in (4), is combined with a proprietary dose planning program, and the resulting treatment plan is delivered by the Linac.

2.4 Making a Sphere by Arcs of Beams

By varying the angle of the gantry and the angle of the table, one can deliver a radiation beam to the target from any angle within the range of the rotation. The shape of the common intersection of an arc of beams passing through one isocenter is a sphere. The neurosurgeons deliver a series of arcs (usually 5 or 9 arcs) to produce a single isocentered sphere shape. For an ellipsoidal target, they use fewer numbers of arcs to make a single isocentered ellipsoid shape. (See Figure 2-3, 2-4 and 2-5.)

So, if the target shape is very close to a sphere or an ellipsoid, then the treatment plan is relatively easy compared to an irregularly shaped target. In that case, we need to create a geometric treatment plan.

2.5 Sphere Packing Plan

As seen in the previous section, the physicians know how to irradiate-to-(eventually)destroy tumors which are shaped like spheres or ellipsoids. For a non-spherical shape of tumor, they try to fill the target with several spheres of different sizes. This is called the 'sphere packing' treatment plan.

After finding a sphere packing plan, they treat each sphere separately as described in the previous section. So, multiple isocenter radiosurgery planning includes

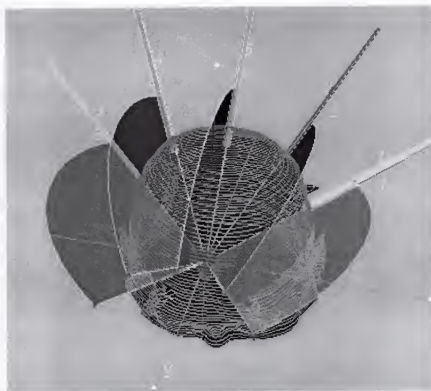


Figure 2-3. One isocenter plan with five arcs of radiation (Wagner et al.[11])

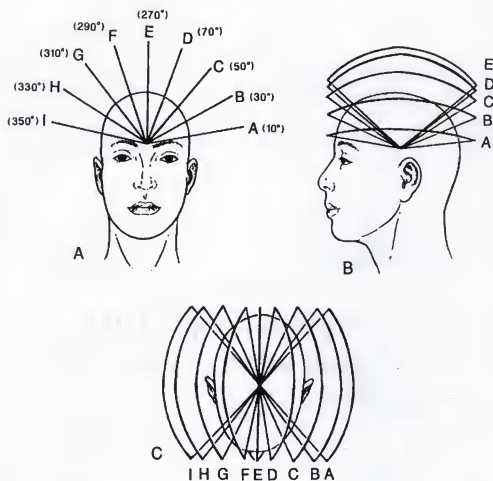


Figure 2-4. One isocenter plan with nine arcs of radiation (Friedman et al.[2])

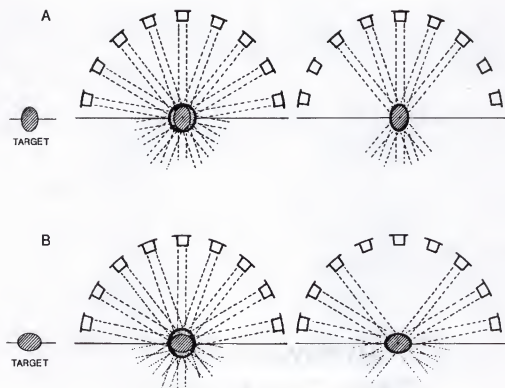


Figure 2-5. Plan for a sphere shape and an ellipsoidal shape (Friedman et al.[2])

the problem of determining the best sphere packing arrangement with which to fill the target volume. General methods for this treatment plan are iteratively based, dosimetrically driven algorithms. But these methods require many computations in order to compute a radiosurgical plan dose distribution, and then to evaluate the quality of the dose distribution. So geometrically based radiosurgery optimization has been suggested as a possible alternative means.

However the method the physicians choose relies on human decisions and experience. Thus, for the same target, different surgeons may produce different plans. Even the same surgeon, doing the plan twice for the same target, may produce different plans. And the planning takes a long time, especially for a complicated target which needs more than 10 spheres. It might take as much as two hours of planning for a difficult case which needs about 20 spheres. During that time, the patient has to wait with the head ring attached to his or her head. And most importantly, even after spending the time to make a plan, many physicians without sufficient experience, are not sure if the plan is a 'good' one.

Therefore, we provide an automated sphere packing method for the treatment plan (see Wagner et al.[11]). This method shows potential to significantly aid the planning of difficult multiple isocenter cases. Based on tests with irregularly shaped phantom targets and with a representative sampling of clinical example cases, the method demonstrates the ability to generate radiosurgery plans comparable to, or of better quality than, multiple isocenter Linac radiosurgery plans found in other literature. At the same time, this program always produces the same treatment plan for the same tumor shape. So it can be used as a 'benchmark' to compare with other plans for the given tumor shape. Moreover, this program provides the treatment plan in a relatively short time. For a very difficult case which needed more than 18 spheres, this program took less than 3 minutes instead of more than the 1.5 hours

which were needed when the physicians created the plan using traditional methods. The algorithm of this program is shown in Figure 2-6.

In the following section, we explain the 'shelling algorithm' which we used in this program to get the centers and sizes of spheres for the sphere packing plan.

2.6 Shelling Algorithm

The shelling procedure is best illustrated in Figure 2-7 to 2-16. The major steps of this shelling procedure are as follows.

(1) Transferring the data of the boundary of the target

From the CT scan, each point which is identified from the target contour data file can be mapped into a three dimensional array, since each CT scan shows the trans-axial view for a specific axial coordinate value. In the three dimensional array, we assign the lateral-, AP- and axial-coordinates to be x-, y- and z-axes, respectively. For each individual case, there are minimum and maximum values for x, y and z-axes. We assume the minimum value is always one. We get this information from the Program for Data Reading. (See Appendix A.1.)

Let the target be minimally enclosed in an x-range beginning with one and ending with M_x , a y-range beginning with one and ending with M_y , and a z-range beginning with one and ending with M_z . Now we put 'safe bumper' layers surrounding the target. That is, we enlarge the box containing the target to $ET = [0, M_x + 1] \times [0, M_y + 1] \times [0, M_z + 1]$. ET stands for 'enlarged target box.' We assume that the dimension of each voxel is $1 \times 1 \times 1$. Then we will number the voxels in a lexicographic manner, beginning with 0 and ending with $(M_x + 1) \times (M_y + 1) \times (M_z + 1) - 1$, giving us a total of $(M_x + 1) \times (M_y + 1) \times (M_z + 1)$ voxels with which we will work.

Now we want to identify the voxels in ET in two different ways: first, by a set of coordinates (a, b, c) , and second, by an integer representing a counting procedure. Here, a is the left hand endpoint of the x-value of that voxel, b is the left hand

endpoint of the y -value of that voxel, and c is the left hand endpoint of the z -value of that voxel. The second notation for the voxel (a, b, c) is by some integer n . The voxels are counted, using a lexicographic ordering beginning with 0 and ending with $(M_x + 1) \times (M_y + 1) \times (M_z + 1) - 1$ which is denoted by *nofdata* in the program (see Appendix A.2). Then the voxel denoted by (a, b, c) is numbered as $c \times (M_x + 1) \times (M_y + 1) + a \times (M_y + 1) + b$. And it is clear that, for every $0 \leq i \leq \text{nofdata}$, there are unique nonnegative integers, a, b, c , such that $i = c \times (M_x + 1) \times (M_y + 1) + a \times (M_y + 1) + b$. That is, for every $0 \leq i \leq \text{nofdata}$, one and only one voxel is assigned. Note that Figure 2-7 to Figure 2-14 show the same trans-axial view at different stages of the algorithm. Thus these figures show the data of a fixed axial-coordinate value.

(2) Assign a status value for each voxel

For every $0 \leq i \leq \text{nofdata}$, the status value for the i -th voxel is denoted by $cc[i]$. That is, we denote the status value for the voxel with coordinates (a, b, c) , by $cc[i]$ where $i = c \times (M_x + 1) \times (M_y + 1) + a \times (M_y + 1) + b$. The following is the procedure to assign status values. First of all, let $cc[i] = -3$ for every $0 \leq i \leq \text{nofdata}$. Then we assign the number, -1 , as the status value for each voxel corresponding to any of the target contour data. Next, for every $0 \leq j \leq M_x$, we assign $cc[j \times (M_x + 1) \times (M_y + 1)] = 0$, because the $j \times (M_x + 1) \times (M_y + 1)$ -th voxel is the voxel at the left lower corner for every j -th trans-axial view; that is, the voxel belongs to the healthy tissue. (See Figure 2-8.)

For each voxel with $cc[i] = -3$, we define $cc[i] = 0$ if any of $cc[i - 1]$, $cc[i + 1]$, $cc[i - (M_y + 1)]$, $cc[i + (M_y + 1)]$ is 0. Note that this method produces zeros for every voxel in the safe bumper layers. (See Figure 2-9.)

(3) Shelling the target voxels

We proceed by mathematical induction on j , for

$$0 \leq j \leq \left\lceil \frac{1}{2} \times \min\{M_x, M_y, M_z\} \right\rceil.$$

For $i = 0$ to $(M_x + 1) \times (M_y + 1) \times (M_z + 1) - 1$, if the i -th voxel is assigned -1 or -3 for its status value, and if any of $cc[i - 1]$, $cc[i + 1]$, $cc[i - (M_y + 1)]$, $cc[i + (M_y + 1)]$, $cc[i - (M_x + 1) \times (M_y + 1)]$, $cc[i + (M_x + 1) \times (M_y + 1)]$ is equal to j , then we assign a new status value for the i -th voxel as $c[i] = j + 1$. Note that Figure 2-7 to Figure 2-14 show only a two-dimensional figure. But this procedure is held in a three-dimensional array. This process is repeated until the deepest lying voxels have been identified. The deepest lying voxel in the entire target volume would be the best location for an isocenter. And at the same time, the status value assigned to the deepest voxel indicates the radius of the sphere, which is one less than the status value of the voxel. This completes our induction. (See Figure 2-10.)

(4) Removing the largest sphere

At this moment, we might have several different voxels which have the same largest status value. So we choose the smallest numbered voxel as the center of the first largest sphere to be removed from the target. But in the program in Wagner et al.[11], we choose the most effective voxel to be the center of the first largest sphere, by using the 'score function' which was developed by Tom Wagner. We think of the resulting sphere in the tumor as 'removed,' and therefore we can assign the new status value zero for these voxels in the sphere. To determine which voxels form the sphere that is removed, we do the following: Let R = voxel at the center of sphere to be removed. Let r = status value of R . Then we remove all voxels S with the property that $\text{distance}(\text{center of } R, \text{center of } S) \leq r - 1$. (See Figures 2-11 and 2-12.)

Now, we proceed by mathematical induction on j , for

$$0 \leq j \leq \left\lceil \frac{1}{2} \times \min\{M_x, M_y, M_z\} \right\rceil.$$

For $i = 0$ to $(M_x + 1) \times (M_y + 1) \times (M_z + 1) - 1$, if the i -th voxel has status value $> j$, we reassign a new status value as follows: if any of $cc[i - 1]$, $cc[i + 1]$, $cc[i - (M_y + 1)]$, $cc[i + (M_y + 1)]$, $cc[i - (M_x + 1) \times (M_y + 1)]$, $cc[i + (M_x + 1) \times (M_y + 1)]$ is equal to j , then we assign a new status value for the i -th voxel as $c[i] = j + 1$. This

process is repeated until the deepest lying voxels have been identified. (See Figures 2-13 to 2-15.)

And we repeat step (4) until we get either 0 or 1 for the status value of each voxel in ET, the enlarged target box. Because the smallest size of the diameter of collimators is 5mm, we finish our program when the status values of all the voxels are zeros and/or ones. (See Figure 2-16.) The remaining part will be taken care of by the 'score function' (see Wagner et al.[11]). After finishing this procedure, we have the spheres filling the target. In Figure 2-17, we see the sphere packing plan within a target by this method.

For further detailed information about the score function and the automated sphere packing plan, we refer the reader to Wagner et al.[11] and Wagner[10].

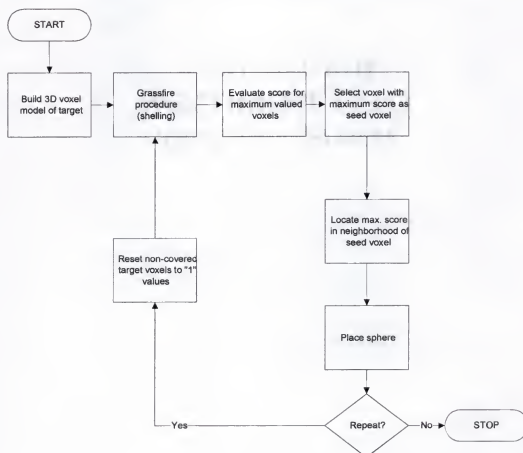


Figure 2-6. Sphere packing algorithm block diagram (Wagner et al.[11])

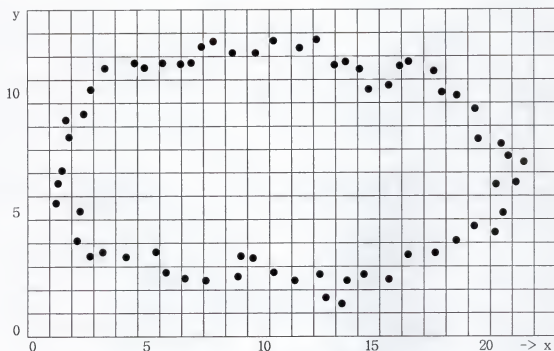


Figure 2-7. Transferring the data of the boundary of the target

-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-1	-1	-1	-1	-1	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-1	-1	-1	-1	-1	-3	-3	-3	-3	-1	-1	-1	-1	-1	-3	-3	-3	-3	-3	-3
-3	-3	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-1	-1	-3	-1	-1	-3	-3	-3	-3
-3	-1	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-3	-3
-3	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-1	-1	-3	-3
-3	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-1	-1	-3
-3	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-1	-1	-3
-3	-1	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3
-3	-3	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-1	-1	-1	-3	-3
-3	-3	-1	-1	-1	-1	-3	-3	-3	-1	-1	-3	-3	-3	-3	-3	-1	-1	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-1	-1	-1	-1	-3	-1	-1	-1	-1	-1	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
0	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3

Figure 2-8. Status values for the boundary and the initial voxel

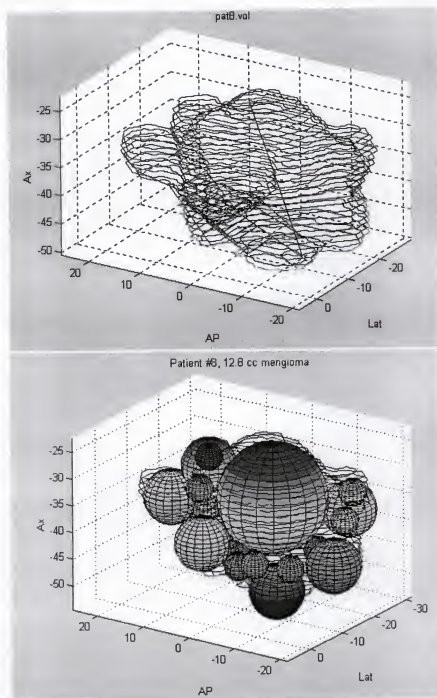


Figure 2-17. A sphere packing plan for a brain tumor (Wagner et al.[11])

CHAPTER 3 A CLASSIFICATION OF UNLABELED TREES

3.1 Introduction

In this chapter, we obtain a code to denote an unlabeled tree. By means of this code, we classify unlabeled n -trees. One may take an unlabeled n -tree to be the geometric realization of a connected, finite, simplicial complex without any simple closed curves. In particular, we call a tree an n -tree if and only if it is a tree with n vertices (that is, $(n - 1)$ edges). Our code assigns a unique, ordered, 'perfect sequence', $pf(T) = \langle d_1, d_2, \dots, d_n \rangle$, to each unlabeled n -tree, T . And, conversely, given an ordered sequence of n integers satisfying certain properties, it is the perfect sequence of exactly one unlabeled n -tree. Thus, we have our main theorem, the Tree Classification Theorem (Theorem 3.5.5). We then show how to reconstruct the tree shape from a given perfect sequence, by hand.

Our work includes an algorithm and a *Mathematica* program (see Appendix C.1) that produce a list of all the perfect sequences for all possible n -trees, thus also producing the number of n -trees, for any given n . For some examples, see Appendix C.2. Since our algorithm and program are based upon our main theorem, we know that all possible unlabeled n -trees are produced by the program. Thus the program gives us, for each positive integer n , a complete list of all unlabeled n -trees.

Given $n > 0$, with sufficient computer power and time, our program will list all the unlabeled n -trees, thus telling us how many unlabeled n -trees there are. However, we do not have a *simple formula* that tells us how many unlabeled n -trees there are. This remains an open problem.

The Literature contains work regarding the classification of *labeled* graphs and *labeled* trees. In Lovász[6], the Prüfer-code is used to determine the number of labeled n -trees (see the Appendix D.2 and D.3). In Sloane[8], Sloane shows a table for $n \leq 26$. However, Sloane references Harary for the method he uses. In Harary[5], there is a simple formula for the total number of *labeled* graphs with n vertices (see the Appendix D.1). There is also a formula for the total number of *unlabeled* n -trees, but it requires knowing the number of *rooted* n -trees, which is calculated inductively. And, even though Harary[5] contains n -tree shapes up to $n = 10$, there is no explanation of how to get the shapes of these trees, while our work does show how to get these tree shapes from our code.

This chapter is organized as follows:

Section 1: Introduction

Section 2: Definitions

Section 3: Vertex Sequence

Section 4: Degree Sequence

Section 5: Perfect Sequence and Main Theorem

Section 6: Obtaining Perfect Sequences

Section 7: Extended Partition and the Algorithm

3.2 Definitions

Most of the necessary definitions in this section are from Harary[5]. We refer the reader to Harary[5] for easy reference.

Definition 3.2.1 A graph $G = (V(G), E(G))$ consists of a vertex set $V(G) = \{v_1, \dots, v_n\}$ and an edge set $E(G) = \{e_1, \dots, e_m\} \subset \{\{u, v\} | u, v \in V(G)\}$ in which each edge is an unordered pair of two distinct vertices u and v , and the edge $\{u, v\}$ is said to join u and v . For an edge $\{u, v\}$ we say that u and v are *adjacent vertices* and the vertex u and the edge $\{u, v\}$ are *incident* with each other, as are

v and $\{u, v\}$. From now on, $|V(G)|$ denotes the number of vertices and $|E(G)|$ denotes the number of edges. For a vertex $u \in V(G)$, the vertex set

$$A(u) = \{v \in G \mid v \text{ is an adjacent vertex of } u\}$$

is called the *adjacent set* of u . Note that $u \notin A(u)$.

Definition 3.2.2 A *walk* is an alternating sequence $\langle v_0, e_1, v_1, e_2, \dots, e_k, v_k \rangle$ of vertices and edges s.t. $v_{i-1} \neq v_i$ for $i = 1, 2, \dots, k$, and $e_i = \{v_{i-1}, v_i\}$ for $i = 1, 2, \dots, k$. If there is no confusion, a walk can be denoted by a sequence of vertices only in the path like $\langle v_1, v_2, \dots, v_k \rangle$. A *cycle* is a walk $\langle v_0, e_1, v_1, e_2, \dots, e_k, v_k \rangle$ with $k \geq 3$ in which $v_0 = v_k$ is the only vertex repeated. A *path* is a walk with no repeated vertex. A (u, v) -*path* is a path with the first vertex u and the last vertex v . The *length* of the (u, v) -path, $l(u, v)$, in a graph is the number of edges in the (u, v) -path. The *distance* $d(u, v)$ between two vertices u and v is the length of a shortest path joining them, if any; otherwise $d(u, v) = \infty$. Note that $d(u, u) = 0$. A graph G is *connected* iff every pair of vertices are joined by a path. A graph having no cycle is *acyclic*. Note that an acyclic graph can be disconnected. A *forest* is an acyclic graph. A *tree* is a connected acyclic graph. An n -*tree* is a tree with n vertices.

Definition 3.2.3 We say that a graph G is *isomorphic* to a graph H if and only if there exists a bijection $f : V(G) \rightarrow V(H)$ such that $\{u, v\} \in E(G)$ if and only if $\{f(u), f(v)\} \in E(H)$.

Definition 3.2.4 An *unlabeled tree* is an isomorphism class of trees.

We are interested in the total number of unlabeled n -trees and their shapes. For example, $T_1 = (\{1, 2, 3\}, \{\{1, 2\}, \{2, 3\}\})$ and $T_2 = (\{1, 2, 3\}, \{\{2, 1\}, \{1, 3\}\})$ are different trees, but there is an isomorphism $f : V(T_1) \rightarrow V(T_2)$ by $f(1) = 2$, $f(2) = 1$, $f(3) = 3$. Thus they are the same unlabeled tree, so we want to count them as one unlabeled 3-tree. From now on, a tree implies an unlabeled tree unless otherwise specified.

Definition 3.2.5 The *degree* of a vertex v , defined by $d(v)$, is the sum of the number of edges incident with v . A vertex v is called an *endvertex* if $d(v) = 1$.

We will use the geometric, that is, topological realization of a graph interchangeably with 'graph' as defined above.

Proposition 3.2.6 [5] *The following statements are equivalent for a graph T :*

- (1) T is a tree.
- (2) Every two vertices of T are joined by a unique path.
- (3) T is connected and $|V(T)| = |E(T)| + 1$.
- (4) T is acyclic and $|V(T)| = |E(T)| + 1$.

Note that there are exactly n vertices and $n-1$ edges in an n -tree.

Proposition 3.2.7 (Degree sum formula) [5] *If G is a graph with the vertex degrees $d_1, \dots, d_n$, then $\sum_{v \in V(G)} d_i = 2|E(G)|$.*

Corollary 3.2.8 *For an n -Tree T we have*

$$\sum_{i=1}^n d(v_i) = 2(n-1) = 2|E(T)| = 2(|V(T)| - 1)$$

where v_i is a vertex of T for $i = 1, 2, \dots, n$.

Because a nontrivial tree satisfies $\sum_{v \in V(T)} d(v) = 2|E(T)| = 2(|V(T)| - 1)$, there are at least two vertices with degree 1. Thus we have the following theorem as a corollary.

Theorem 3.2.9 [5] *Every nontrivial tree has at least two endvertices.*

3.3 Vertex Sequence

In this section, we define the *vertex sequence* of a given n -tree.

Definition 3.3.1 Let $T = (V, E)$ be an n -tree. If $d(v) = m$, then there are m components of $T - \{v\}$. A *branch* of v (or simply a *v -branch*) is a subtree which is the closure of a component of $T - \{v\}$. We call the set of vertices in a v -branch a *v -branch vertex set*. If $u \in A(v)$, then the v -branch vertex set containing u is called the *(v, u) -branch vertex set* and denoted by $B(v, u)$ (or $B(u)$). We call the

set $B'(v, u) = B(v, u) - \{v\}$ the (v, u) -subbranch vertex set, which is denoted by $B'(u)$ if there is no confusion. The subtree consisting of $B'(v, u)$ as the vertex set and $\{\{s, t\} \in B'(v, u) \times B'(v, u) \mid \{s, t\} \in E(T)\}$ as the edge set is called the (v, u) -subbranch.

Example 3.3.2 Let T be the 10-tree shown below.

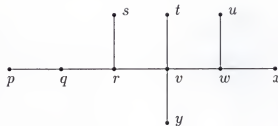


Figure 3-1

Since $d(v) = 4$, there are 4 components of $T - \{v\}$. Since $A(v) = \{r, t, w, y\}$, we have that

$$\begin{array}{ll} B(v, r) = \{v, r, s, q, p\}, & B'(v, r) = \{r, s, q, p\} \\ B(v, t) = \{v, t\}, & B'(v, t) = \{t\} \\ B(v, w) = \{v, w, u, x\}, & B'(v, w) = \{w, u, x\} \\ B(v, y) = \{v, y\}, & B'(v, y) = \{y\}. \end{array}$$

On the other hand, we have that

$$B(r, v) = \{r, v, t, u, w, x, y\}.$$

That is, $B(v, r) \neq B(r, v)$. And this is true for any tree except the 2-tree. In fact, for every n -tree with $n > 2$, there are at least 3 vertices, namely u, v , and w . Without loss of generality, let u and v be vertices adjacent to each other, and let the (w, v) -path contain the vertex u . Then $w \in B(v, u)$ and $w \notin B(u, v)$.

Definition 3.3.3 A *section* of a sequence is a subsequence consisting of consecutive terms of the sequence.

Definition 3.3.4 Let $T = (V, E)$ be an n -tree with $V = \{x_1, x_2, \dots, x_n\}$. We define the set, S , of sequences of the form $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$, where $\hat{s}$ is a permutation of the x_i 's in $V(T)$, to be those sequences satisfying:

- (*) for any v_j with $1 \leq j < n$, $v_{j+1} \in A(v_j) - \{v_1, v_2, \dots, v_j\}$
 where $k = \max\{i \in \{1, 2, \dots, j\} \mid A(v_i) - \{v_1, v_2, \dots, v_j\} \neq \emptyset\}$.

Then S is called the *vertex sequence set* of T and each element, $\hat{s}$, in S is called a *vertex sequence* of T .

Definition 3.3.5 For any $v \in V$, we call $S_v = \{\hat{s} \in S \mid v_1 = v\} \subset S$ the *v-vertex sequence set*.

Definition 3.3.4 requires that the vertices in a sequence be listed in a particular way. The condition (*) ensures that all vertices are listed and that each is listed exactly once. We can choose any vertex for the first term, v_1 . Then the condition (*) says that, for the $(i+1)$ -th term, we have to choose an unlisted adjacent vertex of v_i , if possible. If every adjacent vertex of v_i is already used, then look backward along the sequence until we come to the first vertex which has an unused adjacent vertex. Then, for the $(i+1)$ -th term, we assign one of the unused adjacent vertices of that vertex. To get a vertex sequence, we continue this procedure inductively.

Note that every vertex sequence of an n -tree has n entries.

Example 3.3.6 We show how to construct a vertex sequence of a given tree. Let T be the 7-tree shown below.



Figure 3-2

Since we have freedom to choose the first term, let $v_1 = b$. Then $v_2 \in A(v_1) - \{v_1\} = A(b) - \{b\} = \{a, c, f\} - \{b\} = \{a, c, f\} = A(b) = A(v_1)$. So v_2 must be one of the elements of $A(v_1) = A(b) = \{a, c, f\}$. Let $v_2 = c$. Since $v_3 \in A(v_2) - \{v_1, v_2\} = A(c) - \{b, c\} = \{b, d, g\} - \{b, c\} = \{d, g\}$, v_3 must be either d or g . Let $v_3 = g$. Then $v_4 \notin A(v_3) - \{v_1, v_2, v_3\}$, since $A(v_3) - \{v_1, v_2, v_3\} = \{c\} - \{b, c, g\} = \emptyset$. On

the other hand, we have that $A(v_2) - \{v_1, v_2, v_3\} = \{b, d, g\} - \{b, c, g\} = \{d\} \neq \emptyset$. Therefore we have to choose d as v_4 . Then $v_5 = e$ since $A(v_4) - \{v_1, v_2, v_3, v_4\} = \{c, e\} - \{b, c, g, d\} = \{e\}$. Since we have

$$\begin{aligned} A(v_5) - \{v_1, v_2, v_3, v_4, v_5\} &= \{d\} - \{b, c, g, d, e\} = \emptyset, \\ A(v_4) - \{v_1, v_2, v_3, v_4, v_5\} &= \{c, e\} - \{b, c, g, d, e\} = \emptyset, \\ A(v_3) - \{v_1, v_2, v_3, v_4, v_5\} &= \{c\} - \{b, c, g, d, e\} = \emptyset, \\ A(v_2) - \{v_1, v_2, v_3, v_4, v_5\} &= \{b, g, d\} - \{b, c, g, d, e\} = \emptyset, \text{ and} \\ A(v_1) - \{v_1, v_2, v_3, v_4, v_5\} &= \{a, c, f\} - \{b, c, g, d, e\} = \{a, f\}, \end{aligned}$$

v_6 must be either a or f . Let $v_6 = f$. Then we have that

$$\begin{aligned} A(v_6) - \{v_1, v_2, v_3, v_4, v_5, v_6\} &= \{b\} - \{b, c, g, d, e, f\} = \emptyset, \\ A(v_5) - \{v_1, v_2, v_3, v_4, v_5, v_6\} &= \{d\} - \{b, c, g, d, e, f\} = \emptyset, \\ A(v_4) - \{v_1, v_2, v_3, v_4, v_5, v_6\} &= \{c, e\} - \{b, c, g, d, e, f\} = \emptyset, \\ A(v_3) - \{v_1, v_2, v_3, v_4, v_5, v_6\} &= \{c\} - \{b, c, g, d, e, f\} = \emptyset, \\ A(v_2) - \{v_1, v_2, v_3, v_4, v_5, v_6\} &= \{b, g, d\} - \{b, c, g, d, e, f\} = \emptyset, \text{ and} \\ A(v_1) - \{v_1, v_2, v_3, v_4, v_5, v_6\} &= \{a, c, f\} - \{b, c, g, d, e, f\} = \{a\}. \end{aligned}$$

Therefore v_7 must be a ; that is, we construct a vertex sequence

$$\langle v_1, v_2, \dots, v_7 \rangle = \langle b, c, g, d, e, f, a \rangle.$$

For a given tree we will get different vertex sequences if we choose different vertices whenever we have two or more choices. But the sequence $t = \langle b, c, d, e, f, a, g \rangle$ cannot be a vertex sequence of T above. In fact, we have $v_5 = f$. But we have that

$$\begin{aligned} A(v_4) - \{v_1, v_2, v_3, v_4\} &= \{d\} - \{b, c, d, e\} = \emptyset, \\ A(v_3) - \{v_1, v_2, v_3, v_4\} &= \{c, e\} - \{b, c, d, e\} = \emptyset, \\ A(v_2) - \{v_1, v_2, v_3, v_4\} &= \{b, d, g\} - \{b, c, d, e\} = \{g\}, \text{ and} \\ A(v_1) - \{v_1, v_2, v_3, v_4\} &= \{a, c, f\} - \{b, c, d, e\} = \{a, f\}. \end{aligned}$$

That is, the sequence t has f , one of the elements of $A(v_1) - \{v_1, v_2, v_3, v_4\}$, for v_5 instead of g which is an element of $A(v_2) - \{v_1, v_2, v_3, v_4\}$. It is a contradiction to the maximality of k in the definition of vertex sequence.

Proposition 3.3.7 *Let $\langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of an n -tree T . For any v_j with $j > 1$ there is unique $1 \leq k < j$ such that $v_j \in A(v_k) - \{v_1, v_2, \dots, v_{j-1}\}$.*

Proof First of all, for any $j > 1$, we have $v_j \neq v_i$ for $i = 1, 2, \dots, j-1$ from the Definition 3.3.4. Assume that there is $j > 1$ such that

$$\{v_j, v_{j+1}, \dots, v_n\} \cap \left(\left(\bigcup_{i=1}^{j-1} A(v_i) \right) \cup \{v_1, v_2, \dots, v_{j-1}\} \right) = \emptyset.$$

Then, for any $u \in \{v_j, v_{j+1}, \dots, v_n\}$ and any $w \in (\cup_{i=1}^{j-1} A(v_i)) \cup \{v_1, v_2, \dots, v_j\}$, there is no (u, w) -path in T . That is, T is not connected. It is a contradiction to the connectedness of a tree. Therefore there is $1 \leq k < j$ such that $v_j \in A(v_k) - \{v_1, v_2, \dots, v_{j-1}\}$. If there are $1 \leq k_1 < k_2 < j$ such that $v_j \in A(v_{k_1}) - \{v_1, v_2, \dots, v_{j-1}\}$ and $v_j \in A(v_{k_2}) - \{v_1, v_2, \dots, v_{j-1}\}$, then the two edges, $\{v_j, v_{k_1}\}$ and $\{v_j, v_{k_2}\}$, are elements of $E(T)$. There is a path between v_{k_1} and v_{k_2} in the subtree $\{\{v_1, v_2, \dots, v_{j-1}\}, E'\}$ where $E' = \{\{u, v\} \in E(T) \mid u, v \in \{v_1, v_2, \dots, v_{j-1}\}\}$. The tree T contains a cycle consisting of $\{v_j, v_{k_1}\}$, $\{v_j, v_{k_2}\}$ and the (v_{k_1}, v_{k_2}) -path. This contradicts the fact that a tree is acyclic. Therefore there is only one $1 \leq k < j$ such that $v_j \in A(v_k) - \{v_1, v_2, \dots, v_{j-1}\}$. $\perp$

From the Definition 3.3.4 and the previous lemma we get the following corollary immediately.

Corollary 3.3.8 *Let $\langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of an n -tree. Then*

- (1) $v_2 \in A(v_1)$
- (2) If $d(v_j) \geq 2$ for any $j = 2, 3, \dots, n-1$, then $v_{j+1} \in A(v_j)$.
- (3) For any $j = 2, 3, \dots, n-1$ if $d(v_j) = 1$ and there is $i < j$ such that $d(v_i) \geq 3$ and $d(v_k) = 2$ for every $k = i+1, i+2, \dots, j-1$, then $v_{j+1} \in A(v_i)$ and $v_{j+1} \notin A(v_p)$, for every $p = 1, 2, \dots, i-1, i+1, \dots, j$.

Proposition 3.3.9 *Let $\langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of an n -tree and $A(v_1) = \{v_{1_1}, v_{1_2}, \dots, v_{1_m}\}$ with $1 < 1_1 < 1_2 < \dots < 1_m$. Then*

$$\{v_1, v_2, \dots, v_n\} = \cup_{i=1}^m B(v_1, v_{1_i}).$$

Proposition 3.3.10 *Let $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of a given n -tree T . If $d(v_j) = m$ for some $j > 1$, then there exist m vertices, $v_i, v_{j_1}, v_{j_2}, \dots, v_{j_{m-1}}$ such that $A(v_j) = \{v_i, v_{j_1}, v_{j_2}, \dots, v_{j_{m-1}}\}$ with $i < j < j_1 < j_2 < \dots < j_{m-1}$.*

Proof For any $j > 1$ there exists unique $i < j$ such that $\{v_i, v_j\} \in E(T)$ by Proposition 3.3.7; that is, $v_i \in A(v_j)$. So, for any $v_k \in A(v_j) - \{v_i\}$, $j < k$.

Since $d(v_j) = m$, there are $m - 1$ vertices, $v_{j_1}, v_{j_2}, \dots, v_{j_{m-1}}$, such that $A(v_j) = \{v_i, v_{j_1}, v_{j_2}, \dots, v_{j_{m-1}}\}$ with $i < j < j_1 < j_2 < \dots < j_{m-1}$. $\perp$

Proposition 3.3.11 *Let $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of a given n -tree T . Let $d(v_1) = m$ and $A(v_1) = \{v_{1_1}, v_{1_2}, \dots, v_{1_m}\}$ be the adjacent set of v_1 , named so that $1 < 1_1 < 1_2 < \dots < 1_m$. Then, for any $i = 1, 2, \dots, m$, the set of vertices in the subbranch, $B'(v_1, v_{1_i})$, forms a section in $\hat{s}$ starting with v_{1_i} .*

Proof Clearly, the section containing v_{1_i} starts with $v_{1_i} = v_2$. Let $B'(v_1, v_{1_i}) = \{v_{1_i}, v_{1_i+1}, \dots, v_{1_i+m_1}\}$. Assume that there is a vertex $u \in V(T) - B'(v_1, v_{1_i})$ such that $\langle v_{1_i}, v_{1_i+1}, \dots, v_{1_i+j}, u, \dots \rangle$ is a section of $\hat{s}$. Since $u \notin B'(v_1, v_{1_i})$, $A(v_{1_i+k}) - B'(v_1, v_{1_i}) = \emptyset$ for any $k = 0, 1, 2, \dots, j$ from the definition. Thus $\bigcup_{k=0}^j A(v_{1_i+k}) - B'(v_1, v_{1_i}) = \emptyset$. That is, $\bigcup_{k=0}^j A(v_{1_i+k}) \subset B'(v_1, v_{1_i})$. On the other hand, $A(v_{1_i+k}) \subset B'(v_1, v_{1_i})$ for every $k = 0, 1, 2, \dots, j$. Thus we have that $\bigcup_{k=0}^j A(v_{1_i+k}) \subset B'(v_1, v_{1_i})$. Therefore $\bigcup_{k=0}^j A(v_{1_i+k}) = B'(v_1, v_{1_i})$. That is, $j = m_1$. Hence $B'(v_1, v_{1_i})$ forms a section in $\hat{s}$. Since $B'(v_1, v_{1_i})$ is a subbranch of v_1 , $v_{1_i+m_1+1} \notin A(u)$ for any $u \in B'(v_1, v_{1_i})$. Thus $v_{1_i+m_1+1} \in A(v_i)$. That is, $v_{1_i+m_1+1} = v_{1_2}$. For any $i = 1, 2, \dots, m - 1$, let $B'(v_1, v_{1_i})$ form a section in $\hat{s}$ and $v_{1_i+m_1+1} = v_{1_{i+1}}$. By a similar method, $B'(v_1, v_{1_{i+1}})$ forms a section in $\hat{s}$ and $v_{1_{i+1}+m_{i+1}+1} = v_{1_{i+2}}$. $\perp$

The section formed by a subbranch $B'(v_1, v_{1_i})$ is denoted by $\langle B'(v_1, v_{1_i}) \rangle$. From the previous proposition, we have the following corollaries.

Corollary 3.3.12 *Let $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of a given n -tree T . Let $d(v_1) = m$ and $A(v_1) = \{v_{1_1}, v_{1_2}, \dots, v_{1_m}\}$ be the adjacent set of v_1 , named so that $1 < 1_1 < 1_2 < \dots < 1_m$. The section $\langle B'(v_1, v_{1_i}) \rangle$ precedes the section $\langle B'(v_1, v_{1_{i+1}}) \rangle$ in $\hat{s}$ for $i = 1, 2, \dots, m - 1$, and there is no other vertex between these two sections.*

Corollary 3.3.13 *Let $\langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of a given n -tree T . For any $j > 1$ with $d(v_j) = m \geq 2$ let $A(v_j) = \{v_i, v_{j_1}, v_{j_2}, \dots, v_{j_{m-1}}\}$ where $i <$*

$j < j_1 < j_2 < \dots < j_{m-1}$. Then $\langle B'(v_j, v_{j_i}) \rangle = \langle v_{j_i}, v_{j_i+1}, \dots, v_{j_{(i+1)}-1} \rangle$ for any $i = 1, 2, \dots, m-1$. In particular, we have that $j_1 = j + 1$.

Note that $B(v_j, v_{j_{m-1}}) = \{v_j, v_{j_{m-1}}, v_{j_{m-1}+1}, \dots, v_{j_m-1}\}$ is the $(v_j, v_{j_{m-1}})$ -branch vertex set for some $j_m > j_{m-1}$.

Corollary 3.3.14 Let $\langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of a given n -tree. If $i < j$ with $v_j \in A(v_i)$ and $v_j \notin A(v_k)$ for every $k = i+1, i+2, \dots, j-1$, then $v_p \notin A(v_k)$ for every $p \geq j$ and $k = i+1, i+2, \dots, j-1$.

For any given vertex sequence $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$ and any vertex $v_j \neq v_1$, v_1 belongs to a branch of v_j . We want to give a special name to this branch.

Definition 3.3.15 Let $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of a given n -tree T . For any $j > 1$ the *root branch* of v_j in $\hat{s}$ is the branch of v_j containing v_1 .

Note that there is no root branch for v_1 .

Proposition 3.3.16 Let $\langle v_1, v_2, \dots, v_n \rangle \in S$ be the vertex sequence set of a given n -tree T . For any $1 < j < n$, if $v_{j+1} \notin A(v_j)$, then v_j is an end vertex.

Proof If $v_{j+1} \notin A(v_j)$ for some $1 < j < n$, then v_j has a unique edge with a vertex, v_i , with $i < j$ by Proposition 3.3.7. That is, $d(v_j) = 1$. Therefore v_j is an end vertex. $\perp$

Proposition 3.3.17 Let $\langle v_1, v_2, \dots, v_n \rangle \in S$ be a vertex sequence set of a given n -tree T . For any $1 < j < n$ there is an integer j_m such that

$$\{v_1, v_2, \dots, v_j\} \cup \{v_{j_m}, v_{j_m+1}, \dots, v_n\}$$

is the vertex set of the root branch of v_j .

Proof Let $1 < j < n$ and $d(v_j) = m$. Then, by Proposition 3.3.10, there exist m different vertices, $v_i, v_{j_1}, v_{j_2}, \dots, v_{j_{m-1}}$ such that $A(v_j) = \{v_i, v_{j_1}, v_{j_2}, \dots, v_{j_{m-1}}\}$ with $i < j < j_1 < j_2 < \dots < j_{m-1}$. From Corollary 3.3.13, we have that

$$\langle \bigcup_{k=1}^{m-1} B'(v_j, v_{j_k}) \rangle = \langle v_{j_1}, v_{j_1+1}, \dots, v_{j_m-1} \rangle.$$

That is, the union of all the subbranches of v_j except the root branch is a section from v_{j_1} to v_{j_m-1} . Therefore the set of vertices in the root branch is $\{v_1, v_2, \dots, v_j\} \cup \{v_{j_m}, v_{j_m+1}, \dots, v_n\}$. $\perp$

Note that, if $j_m > n$, then the root branch is $\{v_1, v_2, \dots, v_j\}$.

Corollary 3.3.18 *Let $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle \in S$ be the vertex sequence set of a given n -tree T . Let $j \neq 1$. Then $d(v_j) = 1$, that is, v_j is an end vertex of T , if and only if the set of vertices of the root branch of v_j is $V(T) = \{v_1, v_2, \dots, v_n\}$.*

Proposition 3.3.19 *Let $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of a given n -tree T . For any vertex v_j with $j > 1$ the (v_1, v_j) -path in T is a subsequence of $\langle v_1, v_2, \dots, v_j \rangle$.*

Proof For any vertex v_j with $j > 1$ there is a unique vertex v_{k_p} with $k_p < j$ such that $v_j \in A(v_{k_p})$. For the vertex v_{k_p} there is another unique vertex $v_{k_{p-1}}$ with $k_{p-1} < k_p$ such that $v_{k_p} \in A(v_{k_{p-1}})$. Then we will have that v_1 is the unique adjacent vertex of a previous vertex v_{k_1} . Therefore the (v_1, v_j) -path is a subsequence $\langle v_1, v_{k_1}, \dots, v_{k_{p-1}}, v_{k_p}, v_j \rangle$ of $\langle v_1, v_2, \dots, v_j \rangle$. $\perp$

Proposition 3.3.20 *For any two vertex sequences $\hat{s}_1$ and $\hat{s}_2$ in the v -vertex sequence set $S_v = \{\hat{s} \in S | v_1 = v\}$ the root branch vertex sets of a vertex $x \neq v_1 = v$ in $\hat{s}_1$ and in $\hat{s}_2$ are same.*

Proof Let $\hat{s}_1 = \langle v_1, v_2, \dots, v_n \rangle$ and $\hat{s}_2 = \langle u_1, u_2, \dots, u_n \rangle$ be two different vertex sequences of a given tree T with $v_1 = u_1$. Let x be an arbitrary vertex in T with $x \neq v_1$. Then there are two integers k and l such that $v_k = x = u_l$. Let $A(x) = \{x_1, x_2, \dots, x_{m-1}, y\}$. Since T is a tree, there is a unique (v_1, x) -path in T . Then one of the elements of $A(x)$ is in the (v_1, x) -path. Without loss of generality, let y be in the (v_1, x) -path. From the proof of Proposition 3.3.17 we have that

$$\begin{aligned} \langle \cup_{p=1}^{m-1} B'(x, x_p) \rangle &= \langle \cup_{p=1}^{m-1} B'(v_k, x_p) \rangle \\ &= \langle v_{k+1}, v_{k+2}, \dots, v_{k+r} \rangle \quad \text{and} \\ \langle \cup_{p=1}^{m-1} B'(x, x_p) \rangle &= \langle \cup_{p=1}^{m-1} B'(u_l, x_p) \rangle \\ &= \langle u_{l+1}, u_{l+2}, \dots, u_{l+r} \rangle \end{aligned}$$

for some r . Therefore we have that

$$\{v_1, v_2, \dots, v_k\} \cup \{v_{k+r+1}, \dots, v_n\} = \{u_1, u_2, \dots, u_l\} \cup \{u_{l+r+1}, \dots, u_n\}.$$

That is, the root branch vertex sets of x in $\hat{s}_1$ and in $\hat{s}_2$ are same. $\perp$

Example 3.3.21 Let T be an 8-tree as shown below.

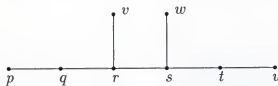


Figure 3-3

Clearly, $\hat{s}_1 = \langle q, r, s, t, u, w, v, p \rangle$, $\hat{s}_2 = \langle q, p, r, s, w, t, u, v \rangle \in S_r = \{\hat{s} \in S | v_1 = q\}$. Let

$$\langle v_1, v_2, \dots, v_8 \rangle = \langle q, r, s, t, u, w, v, p \rangle \text{ and } \langle u_1, u_2, \dots, u_8 \rangle = \langle q, p, r, s, w, t, u, v \rangle.$$

Then the root branch vertex set of s is $\{p, q, r, v\}$. That is, $\{p, q, r, v\} = \{q, r\} \cup \{v, p\} = \{v_1, v_2\} \cup \{v_7, v_8\}$ and $\{p, q, r, v\} = \{q, p, r\} \cup \{v\} = \{u_1, u_2, u_3\} \cup \{u_8\}$. Therefore the root branch vertex sets are same but the position might be different.

On the other hand, $d(w) = 1$ and the unique adjacent vertex of $w = v_6$ is $s = v_3$. Thus $s = v_3$ is in the root branch of $w = v_6$. Therefore the root branch vertex set of $w = v_6$ is $\{v_1, v_2, v_3, v_4, v_5, v_6\} \cup \{v_7, v_8\} = \{v_1, v_2, \dots, v_8\} = \{q, r, s, t, u, w, v, p\}$. That is, the root branch vertex set of an end vertex is $V(T)$ itself.

3.4 Degree Sequence

In this section, we define the degree sequence of a vertex sequence.

Definition 3.4.1 Let S be the vertex sequence set of an n -tree T . For any vertex sequence $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle \in S$, we call $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ the *degree sequence* of $\hat{s}$ where $d_1 = d(v_1)$ and $d_i = d(v_i) - 1$ for $i \geq 2$. We call the set $D = \{D_{\hat{s}} | \hat{s} \in S\}$ the *degree sequence set* of S . And, for any vertex $v \in V$, we

call $D_v = \{D_{\hat{s}} | \hat{s} \in S_v\} \subset D$ the v -degree sequence set of S_v . For any section $\langle B(v_j, v_{j_i}) \rangle = \langle v_j, v_{j_i}, v_{j_i+1}, \dots, v_{j_i+1-1} \rangle$ of $\hat{s}$, formed by the (v_j, v_{j_i}) -branch vertex set $B(v_j, v_{j_i})$, we define $Bd(v_j, v_{j_i}) = \{d_j, d_{j_i}, d_{j_i+1}, \dots, d_{j_i+1-1}\}$ as the (v_j, v_{j_i}) -branch degree set where $d_k = d(v_k) - 1$ for $k = j, j_i, j_i+1, \dots, j_i+1-1$. And we define the (v_j, v_{j_i}) -branch degree section by $\langle Bd(v_j, v_{j_i}) \rangle = \langle d_j, d_{j_i}, d_{j_i+1}, \dots, d_{j_i+1-1} \rangle$ which is denoted by $\langle Bd(v_{j_i}) \rangle$ if there is no confusion. In the same way as in Definition 3.3.1, we define $\langle Bd'(v_{j_i}) \rangle = \langle Bd'(v_j, v_{j_i}) \rangle = \langle d_{j_i}, d_{j_i+1}, \dots, d_{j_i+1-1} \rangle$.

Example 3.4.2 Occasionally, it may happen that there are two or more vertex sequences of a given tree with the same degree sequence. In the following example, there are several pairs of vertex sequences with the same degree sequences, respectively.



Figure 3-4

Since there are 5 vertices in $V(T)$, there are exactly $5!$ different sequences of vertices of T having length 5 and using each vertex only once. But only 16 of them satisfy the condition (*) in Definition 3.3.4. Therefore the vertex sequence set, S , of the given tree, T , consists of the following sequences

$$\begin{aligned} \hat{s}_1 &= \langle p, r, q, s, t \rangle, & \hat{s}_7 &= \langle r, p, q, s, t \rangle, & \hat{s}_{13} &= \langle s, r, p, q, t \rangle, \\ \hat{s}_2 &= \langle p, r, s, t, q \rangle, & \hat{s}_8 &= \langle r, p, s, t, q \rangle, & \hat{s}_{14} &= \langle s, r, q, p, t \rangle, \\ \hat{s}_3 &= \langle q, r, p, s, t \rangle, & \hat{s}_9 &= \langle r, q, p, s, t \rangle, & \hat{s}_{15} &= \langle s, t, r, p, q \rangle, \\ \hat{s}_4 &= \langle q, r, s, t, p \rangle, & \hat{s}_{10} &= \langle r, q, s, t, p \rangle, & \hat{s}_{16} &= \langle s, t, r, q, p \rangle, \\ \hat{s}_5 &= \langle t, s, r, p, q \rangle, & \hat{s}_{11} &= \langle r, s, t, p, q \rangle, \\ \hat{s}_6 &= \langle t, s, r, q, p \rangle, & \hat{s}_{12} &= \langle r, s, t, q, p \rangle. \end{aligned}$$

Then the degree sequences of these vertex sequences are

$$\begin{aligned} D_{\hat{s}_1} &= \langle 1, 2, 0, 1, 0 \rangle, & D_{\hat{s}_7} &= \langle 3, 0, 0, 1, 0 \rangle, & D_{\hat{s}_{13}} &= \langle 2, 2, 0, 0, 0 \rangle, \\ D_{\hat{s}_2} &= \langle 1, 2, 1, 0, 0 \rangle, & D_{\hat{s}_8} &= \langle 3, 0, 1, 0, 0 \rangle, & D_{\hat{s}_{14}} &= \langle 2, 2, 0, 0, 0 \rangle, \\ D_{\hat{s}_3} &= \langle 1, 2, 0, 1, 0 \rangle, & D_{\hat{s}_9} &= \langle 3, 0, 0, 1, 0 \rangle, & D_{\hat{s}_{15}} &= \langle 2, 0, 2, 0, 0 \rangle, \\ D_{\hat{s}_4} &= \langle 1, 2, 1, 0, 0 \rangle, & D_{\hat{s}_{10}} &= \langle 3, 0, 1, 0, 0 \rangle, & D_{\hat{s}_{16}} &= \langle 2, 0, 2, 0, 0 \rangle, \\ D_{\hat{s}_5} &= \langle 1, 1, 2, 0, 0 \rangle, & D_{\hat{s}_{11}} &= \langle 3, 1, 0, 0, 0 \rangle, \\ D_{\hat{s}_6} &= \langle 1, 1, 2, 0, 0 \rangle, & D_{\hat{s}_{12}} &= \langle 3, 1, 0, 0, 0 \rangle. \end{aligned}$$

Now we have that

$$\begin{aligned} D_{s_1} &= D_{s_3}, & D_{s_2} &= D_{s_4}, & D_{s_5} &= D_{s_6}, & D_{s_7} &= D_{s_9}, \\ D_{s_8} &= D_{s_{10}}, & D_{s_{11}} &= D_{s_{12}}, & D_{s_{13}} &= D_{s_{14}}, & D_{s_{15}} &= D_{s_{16}}. \end{aligned}$$

That is, for some different vertex sequences, the degree sequences might be same. And, if the degree sequences are same, then the trees denoted by these degree sequences are isomorphic.

Remark: The sum of the terms in any degree sequence of a given n -tree is the same, namely, $n - 1$. It is 4 in the above example, and it is same as the number of edges in the tree. See Theorem 3.4.6 below. Since $d_1 = d(v_1)$ and $d_i = d(v_i) - 1$ for $i = 2, 3, \dots, n$, we have the following proposition.

Proposition 3.4.3 Let $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ be a degree sequence of a vertex sequence $\hat{s}$ of an n -tree T . Then $0 < d_1 \leq n - 1$, and $0 \leq d_i < n - 1$ for $i = 2, \dots, n$.

Proposition 3.4.4 If $d_j \neq 0$ for any $1 \leq j < n$, then v_{j+1} is an adjacent vertex of v_j .

Proposition 3.4.5 Let $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of an n -tree T and $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ the degree sequence of $\hat{s}$. For any $j \in \{1, 2, \dots, n - 1\}$ with $d_j = m \geq 1$, if the set $\{v_{j+1}, v_{j+2}, \dots, v_k\}$ is the collection of all the v_j -subbranches except the root subbranch, then $k \geq j + m$.

Proof Since $d_j = m = d(v_j) - 1$, from Proposition 3.3.10, the vertex v_j has m subbranches besides the root subbranch. Thus there are at least m endvertices in the collection of those subbranches other than the root branch. Therefore at least m terms in $\langle d_j, d_{j+1}, \dots, d_k \rangle$ are 0. Hence $k \geq j + m$. $\perp$

Theorem 3.4.6 (First Reduced Degree Sum Formula) For any degree sequence $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ of a vertex sequence $\hat{s} \in S$ of a given n -tree T we have

$$\sum_{i=1}^n d_i = n - 1 = |E|$$

and

$$d_1 + \sum_{i=2}^n (d_i - 1) = 0.$$

Proof d_1 is the degree of a vertex, and the remaining $n - 1$ terms are each one less than the degree of a vertex. By the degree sum formula (Proposition 3.2.7), it is true that $\sum_{i=1}^n d(v_i) = 2|E|$. So

$$\begin{aligned}
 \sum_{i=1}^n d_i &= d_1 + \sum_{i=2}^n d_i \\
 &= d(v_1) + \sum_{i=2}^n (d(v_i) - 1) \\
 &= d(v_1) + \sum_{i=2}^n d(v_i) - \sum_{i=2}^n 1 \\
 &= (d(v_1) + \sum_{i=2}^n d(v_i)) - (n - 1) \\
 &= \sum_{i=1}^n d(v_i) - (n - 1) \\
 &= 2|E| - (n - 1) \\
 &= 2(n - 1) - (n - 1) \\
 &= n - 1 = |E|.
 \end{aligned}$$

Then we have that

$$\begin{aligned}
 d_1 + \sum_{i=2}^n (d_i - 1) &= d_1 + \sum_{i=2}^n d_i - \sum_{i=2}^n 1 \\
 &= \sum_{i=1}^n d_i - (n - 1) \\
 &= (n - 1) - (n - 1) \\
 &= 0. \perp
 \end{aligned}$$

Corollary 3.4.7 For any degree sequence, $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ of a vertex sequence $\hat{s} \in S$ of a given n -tree T , we have

$$d_1 + \sum_{i=2, d_i \neq 0}^n (d_i - 1) = \sum_{i=2, d_i=0}^n 1.$$

Proof From the first reduced degree sum formula we have that $d_1 + \sum_{i=2}^n (d_i - 1) = 0$.

Then

$$\begin{aligned}
 0 &= d_1 + \sum_{i=2}^n (d_i - 1) \\
 &= d_1 + \sum_{i=2, d_i \neq 0}^n (d_i - 1) + \sum_{i=2, d_i=0}^n (d_i - 1) \\
 &= d_1 + \sum_{i=2, d_i \neq 0}^n (d_i - 1) + \sum_{i=2, d_i=0}^n (-1) \\
 &= d_1 + \sum_{i=2, d_i \neq 0}^n (d_i - 1) - \sum_{i=2, d_i=0}^n 1.
 \end{aligned}$$

Thus we get the result. $\perp$

Note that $d_i \geq 0$ for any $i = 1, 2, \dots, n$. So $d_i - 1 < 0$ if and only if $d_i = 0$.

Corollary 3.4.7 shows that, in a degree sequence $D \langle d_1, d_2, \dots, d_n \rangle$, the total number of 0 terms is the same as $d_1 + \sum_{i=2, d_i \neq 0}^n (d_i - 1)$. Therefore there are exactly $d_1 + \sum_{i=2, d_i \neq 0}^n (d_i - 1)$ endvertices in the given tree. Because of this, we have the following proposition.

Proposition 3.4.8 For any degree sequence $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ of a vertex sequence $\hat{s} \in S$ of a given n -tree T , we have

$$d_1 + \sum_{i=2}^q (d_i - 1) \neq 0$$

for every $1 < q < n$.

Proof Assume that there is $q < n$ such that $d_1 + \sum_{i=2}^q (d_i - 1) = 0$. Since $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ is a degree sequence of a vertex sequence $\hat{s} \in S$ of the given n -tree T , we have that $v_{q+1} \in A(v_p)$ for some $p \leq q$. Thus at least one of the endvertices counted by v_p is one of $v_{q+1}, v_{q+2}, \dots, v_n$. That is, in the section $\langle d_1, d_2, \dots, d_q \rangle$, we have at least one less 0 terms than $d_1 + \sum_{i=2, d_i \neq 0}^q (d_i - 1)$, that is,

$$d_1 + \sum_{i=2, d_i \neq 0}^q (d_i - 1) - \sum_{i=2, d_i=0}^q 1 \geq 1.$$

Then

$$\begin{aligned} 0 &= d_1 + \sum_{i=2}^q (d_i - 1) \\ &= d_1 + \sum_{i=2, d_i \neq 0}^q (d_i - 1) + \sum_{i=2, d_i=0}^q (d_i - 1) \\ &= d_1 + \sum_{i=2, d_i \neq 0}^q (d_i - 1) + \sum_{i=2, d_i=0}^q (-1) \\ &= d_1 + \sum_{i=2, d_i \neq 0}^q (d_i - 1) - \sum_{i=2, d_i=0}^q 1 \\ &\geq 1. \end{aligned}$$

This is a contradiction. Therefore we have $d_1 + \sum_{i=2}^q (d_i - 1) \neq 0$ for every $1 < q < n$. $\perp$

Theorem 3.4.9 (Second Reduced Degree Sum Formula) Let $T = (V, E)$ be an n -tree. Let $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ be the degree sequence of a vertex sequence $\hat{s} \in S$ of the given n -tree T . For any $j_k > 1$, let $B'(v_{j_k}) = \{v_{j_k}, v_{j_k+1}, \dots, v_{j_k+1-1}\}$ be a (v_j, v_{j_k}) -subbranch vertex set. Then we have

$$d_{j_k} + \sum_{i=j_k+1}^{j_k+1-1} (d_i - 1) = 0$$

and

$$d_{j_k} + \sum_{i=j_k+1}^q (d_i - 1) \neq 0$$

for every $j_k < q < j_{k+1}$.

Proof Let $E(B'(v_{j_k})) = \{\{u, v\} \in E(T) | u, v \in B'(v_{j_k})\}$. Then

$$T(v_{j_k}) = (B'(v_{j_k}), E(B'(v_{j_k}))) \subset T$$

is also a tree, and $\langle B'(v_{j_k}) \rangle = \langle v_{j_k}, v_{j_k+1}, \dots, v_{j_k+1-1} \rangle$ is a vertex sequence of the tree $T(v_{j_k})$. Note that $d(v_{j_k}) = d_{j_k} + 1$ in T . Since $v_j \notin B'(v_{j_k})$, the edge $\{v_j, v_{j_k}\}$ is not in the edge set $E(B'(v_{j_k}))$. Thus we have that $d(v_{j_k}) = d_{j_k}$ in the subtree

$T(v_{j_k})$. Then the result comes from the first reduced degree sum formula (Theorem 3.4.6) and Proposition 3.4.8. $\perp$

Corollary 3.4.10 Let $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$ be a vertex sequence of an n -tree T and $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ the degree sequence of $\hat{s}$. For any $d_j > 0$, if the subsequence $\langle v_j, v_{j+1}, \dots, v_p \rangle$ is the collection of all the v_j -branches except the root subbranch of v_j , then

$$d_j + \sum_{i=j+1}^p (d_i - 1) = 0$$

and

$$d_j + \sum_{i=j+1}^q (d_i - 1) \neq 0$$

for every $j < q < p$.

Proof For $j = 1$, they are true by the first reduced degree sum formula (Theorem 3.4.6) and Proposition 3.4.8. Let $j > 1$ and $v_j \in A(v_t)$ for some $t < j$.

(i) If $d_j = 0$, then $d(v_j) = 1$, that is, v_j is an end vertex. Since $j > 1$, $v_j \neq v_1$. Thus the root branch is the whole tree by Corollary 3.3.18. Therefore $\langle v_j \rangle$ is the collection of all the v_j -branches except the root subbranch and $d_j + \sum_{i=j+1}^p (d_i - 1) = d_j = 0$.

(ii) If $d_j > 0$, then the subsequence $\langle v_j, v_{j+1}, \dots, v_p \rangle$ which is the collection of all the v_j -branches except the root subbranch, is the (v_t, v_j) -subbranch, since $v_j \in A(v_t)$. By the previous theorem, we get $d_j + \sum_{i=j+1}^p (d_i - 1) = 0$. Thus the collection of all the v_j -branches except the root subbranch of v_j is the (v_t, v_j) -subbranch. Therefore, by the second reduced degree sum formula, we get $d_j + \sum_{i=j+1}^q (d_i - 1) \neq 0$ for every $j < q < p$. $\perp$

Not all of the nonnegative integer sequences with n entries can be a vertex sequence of an n -tree. There is a condition to being a degree sequence of an n -tree.

Definition 3.4.11 Let $D = \langle d_1, d_2, \dots, d_n \rangle$ be a nonnegative integer sequence. We say that D satisfies the *degree sequence condition*, briefly *DSC*, if and only if it satisfies all of the following:

(a) $d_1 > 0$.

(b) $\sum_{i=1}^n d_i = n - 1$.

(c) For any $j < n$ with $d_j = m > 0$, there are exactly $m + 1$ integers, $j < j_1 = j + 1 < j_2 < \dots < j_m < j_{m+1} (= n + 1 \text{ if } j = 1)$, such that

$$d_{j_k} + \sum_{i=j_k+1}^{j_{k+1}-1} (d_i - 1) = 0 \quad \text{for } k = 1, 2, \dots, m.$$

(d) For any j_k with $d_{j_k} = l > 0$, there are exactly $l + 1$ integers, $j_k < j_{k_1} = j_k + 1 < j_{k_2} < \dots < j_{k_l} < j_{k_{l+1}} = j_{k+1}$ such that

$$d_{j_{k_p}} + \sum_{i=j_{k_p}+1}^{j_{k_{p+1}}-1} (d_i - 1) = 0 \quad \text{for } p = 1, 2, \dots, l.$$

Theorem 3.4.12 Let $D = \langle d_1, d_2, \dots, d_n \rangle$ be a nonnegative integer sequence. Then D is a degree sequence of a vertex sequence of an n -tree if and only if D satisfies DSC.

Proof $[\Rightarrow]$ This follows from Proposition 3.4.3, Theorem 3.4.6 and Theorem 3.4.9.

$[\Leftarrow]$ Let $V = \{v_1, v_2, \dots, v_n\}$ be a set of n different vertices. Let $D = \langle d_1, d_2, \dots, d_n \rangle$ be a nonnegative integer sequence satisfying the above conditions. Since $d_1 > 0$ and $d_i \geq 0$ for any $i = 2, 3, \dots, n$, we can assign the degree of each vertex $v_i \in V$ by $d(v_1) = d_1$ and $d(v_i) = d_i + 1$ for $i = 2, 3, \dots, n$. Let $j < n$ and $d_j = m > 0$. From (c), there are exactly $m + 1$ integers, $j < j_1 < j_2 < \dots < j_m < j_{m+1}$, such that $j_1 = j + 1$, and $d_{j_k} + \sum_{i=j_k+1}^{j_{k+1}-1} (d_i - 1) = 0$ for $k = 1, 2, \dots, m$. Now we assign an edge between v_j and v_{j_k} for $k = 1, 2, \dots, m$. That is, $v_{j_k} \in A(v_j)$ for $i = 1, 2, \dots, m$.

(i) If $d_{j_k} = l > 0$, we assign an edge between v_{j_k} and $v_{j_{k_p}}$ for $p = 1, 2, \dots, l$ by (d). We do a similar procedure for each vertex, v_j , assigned to $d_j > 0$. Therefore $v_{j_k} \in A(v_j)$ and $v_{j_{k_p}} \in A(v_{j_k})$ for $k = 1, 2, \dots, m$, $j = 1, 2, \dots, l$.

(ii) If $d_{j_i} = 0$, then we do not assign any new edge for the vertex v_{j_i} . But it is already joined to the vertex v_j by an edge.

Let $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$. Then all the vertices in $\hat{s}$ are connected. Since each vertex v_i is assigned d_i new edges and $\sum_{i=1}^n d_i = n - 1$ from (b), we have n vertices and

$n - 1$ edges. Then the set of all the edges assigned above, together with the vertex set V , forms an n -tree by Proposition 3.2.6. $\perp$

Remark: The proof of Theorem 3.4.12 shows how to construct a tree that correspond to the given degree sequence satisfying DSC.

Example 3.4.13 Let $D = \langle 3, 2, 3, 1, 0, 0, 0, 1, 0, 2, 0, 0, 0 \rangle$ be a nonnegative integer sequence having 13 entries. Let $V = \{v_1, v_2, \dots, v_n\}$ be a set of n different vertices.



Figure 3-5

- (a) Clearly, $\sum_{i=1}^{13} d_i = 12$ and $d_1 = 3 > 0$. So we expect a 13-tree.
- (b) Since $d(v_1) = 3$, there are 4 different integers, $1 < l_1 = 2 < l_2 = 10 < l_3 = 13 < l_4 = 14$, such that $d_2 + \sum_{i=3}^9 (d_i - 1) = 2 + (2 + 0 - 1 - 1 - 1 + 0 - 1) = 0$, $d_{10} + \sum_{i=11}^{12} (d_i - 1) = 2 + (-1 - 1) = 0$, and $d_{13} = 0$. We assign edges between v_1 and each of v_2, v_{10}, v_{13} .
- (c) (i) Since $v_2 = 2$, there 3 different integers, $2 < 3 < 8 < 10$, such that $d_3 + \sum_{i=4}^7 (d_i - 1) = 3 + (0 - 1 - 1 - 1) = 0$, and $d_8 + (d_9 - 1) = 1 - 1 = 0$. We assign edges between v_2 and each of v_3 and v_8 .
- (ii) Since $v_{10} = 2$, there are 3 different integers, $10 < 11 < 12 < 13$, such that $d_{11} = 0$ and $d_{12} = 0$. We assign edges between v_{10} and each of v_{11} and v_{12} .
- (iii) Since $v_{13} = 0$, we do not assign any new edge for v_{13} . But the vertex v_{13} is connected to v_1 by an edge.

In a similar way, we get $\{v_3, v_4\}, \{v_3, v_6\}, \{v_3, v_7\}, \{v_8, v_9\}$, and $\{v_4, v_5\}$. Let E be the collection of all the edges above, that is,

$$E = \{\{v_1, v_2\}, \{v_1, v_{10}\}, \{v_1, v_{13}\}, \{v_2, v_3\}, \{v_2, v_8\}, \{v_{10}, v_{11}\}, \\ \{v_{10}, v_{12}\}, \{v_3, v_4\}, \{v_3, v_6\}, \{v_3, v_7\}, \{v_8, v_9\}, \{v_4, v_5\}\}.$$

Since the edge set E contains 12 elements, the set $T = \{V, E\}$ is a 13-tree.

3.5 Perfect Sequence and Main Theorem

In the previous section, we showed that, for a given n -tree T , there exist many vertex sequences depending on the starting vertex and choice among the adjacent vertices for each successive vertex. Therefore the degree sequence set D for the given tree T contains many different degree sequences which denote the same tree T . We need a way to choose one degree sequence representing the tree T . First of all we want to define an order on the set of all finite, nonnegative, integer sequences.

Definition 3.5.1 For any two finite nonnegative integer sequences $D = \langle p_1, p_2, \dots, p_k \rangle$ and $E = \langle q_1, q_2, \dots, q_l \rangle$, $D < E$ if and only if either there is $m \leq \min\{k, l\}$ such that $p_m < q_m$ and $p_i = q_i$ for all $i = 1, 2, \dots, m-1$, or $k < l$ and $p_i = q_i$ for all $i = 1, 2, \dots, k$. We write $D \leq E$ if $D < E$ or $D = E$.

Proposition 3.5.2 Let D be the set of all degree sequences of an n -tree. The degree sequence set, $(D, \leq)$, is a totally ordered finite set.

Proof For any two different degree sequences the order defined above is well defined. Since each term d_i of a degree sequence is nonnegative and less than n and the length of each degree sequence is n , the degree sequence set D is a finite set. Thus $(D, \leq)$ is a totally ordered finite set. $\perp$

Definition 3.5.3 For a given n -tree T , let S be the vertex sequence set of T and D the degree sequence set of T . Then the maximum degree sequence in D is called the *perfect sequence* of the given tree T , and it is denoted by $pf(T)$. For the perfect sequence, $pf(T)$, of a given tree, T , there is a vertex sequence $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle \in$

S such that the degree sequence of $\hat{s}$ is $pf(T)$. We call $\hat{s}$ a *perfect vertex sequence* of the tree and v_1 a *perfect vertex* of the tree.

In the Example 3.4.2, the perfect sequence is $\langle 3, 1, 0, 0, 0 \rangle$ and there are two different perfect vertex sequences, $\hat{s}_{11} = \langle r, s, t, p, q \rangle$ and $\hat{s}_{12} = \langle r, s, t, q, p \rangle$ for it. But the trees represented by $\hat{s}_{11}$ and $\hat{s}_{12}$ are isomorphic.

For any given tree the perfect sequence is unique by Proposition 3.5.2. But there may exist several different perfect vertex sequences. If the given tree is a labeled tree, then these perfect vertex sequences denote different labeled trees. But these labeled trees are isomorphic to each other; that is, there is a unique unlabeled tree. Therefore, for a given perfect sequence, there exists a unique unlabeled tree.

Theorem 3.5.4 *Let $d_{\hat{s}}$ be the perfect sequence of a tree T . If T' is any other tree whose perfect sequence is also $d_{\hat{s}}$, then T' is isomorphic to T .*

Proof If T has more than one vertex sequence with the same degree sequence, then these vertex sequences all produce the isomorphic trees, by using the construction in the proof of Theorem 3.4.12. $\perp$

Because of this theorem, we have our main theorem below.

Theorem 3.5.5 (Tree Classification Theorem) *For any positive integer n , let $\mathcal{T}(n)$ be the set of unlabeled n -trees and $\mathcal{P}(n)$ the set of perfect sequences of n -trees. Then there is a one-to-one correspondence between $\mathcal{T}(n)$ and $\mathcal{P}(n)$.*

3.6 Obtaining Perfect Sequences

By Theorem 3.5.5, if we can construct the set of perfect sequences of n -trees, $\mathcal{P}(n)$, without being given any tree shape, then we can classify all the n -trees by this set. And, in the same way as in Example 3.4.13, we can get the different tree shapes from the elements of $\mathcal{P}(n)$.

Proposition 3.6.1 *For the perfect vertex v_1 of a perfect vertex sequence $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$, we have $d(v_1) \geq d(v_i)$ for $i = 2, 3, \dots, n$.*

Proof If there is some $k \in \{2, 3, \dots, n\}$ such that $d(v_k) > d(v_1)$, then $D_{\hat{s}} = \langle d(v_1), \dots \rangle < D_{\hat{s}'} = \langle d(v_k), \dots \rangle$ for any vertex sequence $\hat{s}' = \langle v_k, \dots \rangle$. This contradicts the maximality of $D_{\hat{s}}$. $\perp$

Proposition 3.6.2 If $D_{\hat{s}} = \langle d_1, d_2, \dots, d_n \rangle$ is the perfect sequence of a given n -tree T , then $n-1 \geq d_1 > d_i \geq d_n = 0$ for any $i = 2, 3, \dots, n-1$.

Proof Since T is an n -tree and $D_{\hat{s}}$ is the perfect sequence, we have $n-1 \geq d_1$. For any vertex $u \in V(T)$ we have $d(v_1) \geq d(u)$ by Proposition 3.6.1. Since $d_1 = d(v_1)$ and $d_i = d(v_i) - 1$ for every $1 < i \leq n$, we have $d_1 > d_i$ for $1 < i \leq n$. Since d_n must denote one of the end vertices, each of which has degree 1, we have $d_n = 0$. Therefore we have $d_i \geq d_n = 0$ for every $1 \leq i \leq n-1$. $\perp$

Definition 3.6.3 Let $D = \langle d_1, d_2, \dots, d_n \rangle$ be a degree sequence. Then we say that D satisfies the *ordered subbranch condition*, denoted *OSbC*, if and only if D satisfies the following condition for all $j > 1$:

If $d(v_j) = m+1 \geq 2$, and $B'(v_{j_1}), B'(v_{j_2}), \dots, B'(v_{j_m})$ with $j < j_1 < j_2 < \dots < j_m$ are the subbranches of the vertex v_j other than the root subbranch, then $\langle Bd'(v_{j_k}) \rangle \geq \langle Bd'(v_{j_{k+1}}) \rangle$ for every $k = 1, 2, \dots, m-1$.

Recall that $\langle Bd'(v_{j_i}) \rangle$ is a section of the degree sequence D . (See Definition 3.3.3 and 3.4.1.)

Remark : If $j = 1$, then there is no root branch of v_1 ; instead, there is one more subbranch, $Bd'(v_{j_{m+1}})$, such that $j_m < j_{m+1}$ and $\langle B'(v_{j_m}) \rangle \geq \langle Bd'(v_{j_{m+1}}) \rangle$.

Theorem 3.6.4 Let $D = \langle d_1, d_2, \dots, d_n \rangle$ be a nonnegative integer sequence. If D is the perfect sequence of an n -tree, then it satisfies the *DSC* and the *OSbC*.

Proof It is clear from Theorem 3.4.12 and the maximality of D . $\perp$

Proposition 3.6.5 Let T be a tree and v a vertex with the largest degree in T . Then there is exactly one degree sequence D_v for T starting at v and satisfying both *DSC* and the *OSbC*.

By the above proposition, the converse of Theorem 3.6.4 is not true. The following example is a counterexample.

Example 3.6.6 Let $A = \langle d_1, d_2, \dots, d_{12} \rangle = \langle 3, 2, 2, 1, 0, 0, 2, 0, 0, 1, 0, 0 \rangle$. Then $\sum_{i=1}^{12} d_i = 11$ and $d_1 = 3 > 0$. And the sequence A can be separated into 4 sections like $\langle 3 \rangle$, $\langle 2, 2, 1, 0, 0, 2, 0, 0 \rangle$, $\langle 1, 0 \rangle$, $\langle 0 \rangle$. The second section can be separated into 3 sections, $\langle 2 \rangle$, $\langle 2, 1, 0, 0 \rangle$, $\langle 2, 0, 0 \rangle$, and the second and third section of these can be separated by $\langle 2 \rangle$, $\langle 1, 0 \rangle$, $\langle 0 \rangle$ and $\langle 2 \rangle$, $\langle 0 \rangle$, $\langle 0 \rangle$, respectively. All of these subsections satisfy the second and third conditions of DSC. Thus the sequence A satisfies the DSC. That is, the sequence A is a degree sequence of a vertex sequence $\langle v_1, v_2, \dots, v_{12} \rangle$ of a 12-tree by Theorem 3.4.12. On the other hand, the sequence A satisfies the OSbC, since we have $\langle 2, 2, 1, 0, 0, 2, 0, 0 \rangle > \langle 1, 0 \rangle > \langle 0 \rangle$. But A is not the perfect sequence of the 12-tree. In fact, we rearrange the sequence A to get $B = \langle d_7 + 1, d_2, d_1 - 1, d_{10}, d_{11}, d_{12}, d_3, d_4, d_5, d_6, d_8, d_9 \rangle = \langle 3, 2, 2, 1, 0, 0, 2, 1, 0, 0, 0, 0 \rangle$. Note that $d(v_1) = d_1$ and $d(v_7) = d_7 + 1$. Clearly the sequence B also satisfies the DSC and the OSbC. So B is also a degree sequence of another vertex sequence $\langle v_7, v_2, v_1, v_{10}, v_{11}, v_{12}, v_3, v_4, v_5, v_6, v_8, v_9 \rangle$ of the 12-tree. That is, A and B are degree sequences of the same 12-tree (see Figure 3-6). And we have

$$B = \langle 3, 2, 2, 1, 0, 0, 2, 1, 0, 0, 0, 0 \rangle > \langle 3, 2, 2, 1, 0, 0, 2, 0, 0, 1, 0, 0 \rangle = A.$$

Thus A is not the perfect sequence. Furthermore A cannot be the perfect sequence of any tree, because A and B denote isomorphic trees, and $B > A$.

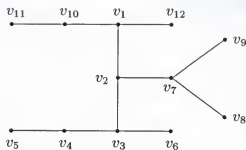


Figure 3-6

In this example we have that $d(v_1) = 3 = d(v_7)$ which is the largest degree in the tree. That is, the perfect sequence must start with the largest degree. We will see in Example 3.6.8 that the degree sequence B is the perfect sequence of this tree.

Note that $\langle 1, 0 \rangle$ is the only possible degree sequence when $d_1 = 1$. Therefore we assume $d_1 > 1$ from now on.

Now we show how to construct alternate degree sequences from A , similar to B . Assume that we have a nonnegative integer sequence $A = \langle d_1, d_2, \dots, d_n \rangle$ satisfying the DSC and the OSbC. If there is $1 < i \leq n$ such that $d_1 \leq d_i$, then A cannot be the perfect sequence of any tree by Proposition 3.6.2. So we assume that $d_1 > d_i$ for $i = 2, 3, \dots, n$, and there is at least one integer $1 < j < n$ such that $d_j = d_1 - 1$. That is, there is an n -tree $T = (V, E)$ such that A is a degree sequence of a vertex sequence $\hat{s} = \langle v_1, v_2, \dots, v_n \rangle$ of T , and T has at least two vertices of maximum degree d_1 . Clearly, $d_j > 0$. Thus v_j is not an end vertex in T . By Corollary 3.3.18, the set of the vertices of the root branch of v_j is not the set of vertices in A itself. So, from Proposition 3.3.17 there is $j < m < n$ such that $RB(v_j) = \{v_1, v_2, \dots, v_j\} \cup \{v_{m+1}, v_{m+2}, \dots, v_n\}$ is the *set of the vertices of the root branch* of v_j . Clearly, $\langle v_{j+1}, v_{j+2}, \dots, v_m \rangle$ is the section containing all of the subbranches of v_j except the root branch. Therefore, if $d_j = p$, then there are p subsections of A like $\langle Bd'(v_{j_1}) \rangle \geq \langle Bd'(v_{j_2}) \rangle \geq \dots \geq \langle Bd'(v_{j_p}) \rangle$ with $j < j_1 < j_2 < \dots < j_p < m+1$, since A satisfies OSbC. On the other hand, let $E(RB'(v_j)) = \{\{u, v\} \in E(T) | u, v \in RB'(v_j)\}$ be the *set of edges in the root subbranch* of v_j . Then $T' = \{RB'(v_j), E(RB'(v_j))\}$ is a subtree of T . Let $S(T')$ be the vertex sequence set of T' . By Proposition 3.3.7, there is a unique integer $1 \leq k < j$ such that $v_k \in A(v_j)$. Clearly, $v_k \in RB'(v_j)$. Let $S_{v_k}(T')$ be the v_k -vertex sequence set of $S(T')$, in which the first term of each vertex sequence is v_k . Let $D_{v_k}(T')$ be the v_k -degree sequence set of $S_{v_k}(T')$ (see Definition 3.4.1). Since $D_{v_k}(T')$ is a finite set, there is a maximum sequence in $D_{v_k}(T')$. We denote the *maximum sequence* in

$D_{v_k}(T')$ by $mdR(v_k)$. Note that d_1 must be changed to $d_1 - 1$ in $mdR(v_k)$, since it is not going to be used for the first term anymore in the new degree sequence. Now we have that either $mdR(v_k) > \langle Bd'(v_{j_1}) \rangle$, or there is an integer $1 \leq q \leq p$ such that $\langle Bd'(v_{j_q}) \rangle \geq mdR(v_k) \geq \langle Bd'(v_{j_{q+1}}) \rangle$. Note that $mdR(v_k)$ may be the smallest section, then $v_{j_{q+1}}$ would not exist.

We rearrange the given degree sequence A to a new sequence in which $d_j + 1$ is the first term and the subbranch degree sequences of v_j ,

$$mdR(v_k), \langle Bd'(v_{j_1}) \rangle, \langle Bd'(v_{j_2}) \rangle, \dots, \langle Bd'(v_{j_p}) \rangle,$$

are listed in descending order. Denote this new degree sequence by $M_{v_j}^A$.

Since the subbranch degree sequences of v_j are listed in descending order in $M_{v_j}^A$, the new degree sequence $M_{v_j}^A$ satisfies the OSbC. Since $M_{v_j}^A$ is a rearrangement of A with $d_1 = d(v_j) = d_j + 1 > 0$, $d_i = d(v_1) - 1 = d_1 - 1$ for some $i > 1$, and A satisfies the DSC, the new degree sequence $M_{v_j}^A$ satisfies the conditions (a) and (b) of the definition of the DSC. Since $mdR(v_k)$ is the perfect sequence of T' , it satisfies the conditions (c) and (d) of the definition of the DSC. Clearly, each section, $\langle Bd'(v_{j_i}) \rangle$, satisfies the conditions (c) and (d) of the definition of the DSC. Therefore the new degree sequence $M_{v_j}^A$ satisfies the DSC. Thus the new degree sequence $M_{v_j}^A$ satisfies both the DSC and the OSbC.

For the given nonnegative integer sequence $A = \langle d_1, d_2, \dots, d_n \rangle$, the set $\{M_{v_j}^A | d(v_j) = d(v_1)\}$ is a non-empty finite set, since $A \in \{M_{v_j}^A | d(v_j) = d(v_1)\}$. Therefore there is a maximum element in this set. Let $P_A = \max\{M_{v_j}^A | d(v_j) = d(v_1)\}$. Now A determines a unique tree T . Suppose B is another vertex sequence for T satisfying the DSC and the OSbC. Then $\{M_{v_q}^B | d(v_q) = d(v_1)\} = \{M_{v_j}^A | d(v_j) = d(v_1)\}$, where v_q is an analogue of v_j , by Proposition 3.6.5. Therefore P_A is the perfect sequence for T .

By the above explanation together with Proposition 3.6.2 and Theorem 3.4.12, we have the following theorem.

Theorem 3.6.7 Let $A = \langle d_1, d_2, \dots, d_n \rangle$ be a nonnegative integer sequence satisfying the DSC and the OSbC with $d_1 > d_i$ for $i = 2, 3, \dots, n$. Then $A = P_A$ if and only if A is the perfect sequence of an n -tree.

In the following example, we show the degree sequence B in Example 3.6.6 is the perfect sequence of the tree.

Example 3.6.8 In Example 3.6.6,

$$A = \langle d_1, d_2, \dots, d_{12} \rangle = \langle 3, 2, 2, 1, 0, 0, 2, 0, 0, 1, 0, 0 \rangle$$

satisfies the DSC and the OSbC and $d_2 = d_3 = d_7 = d_1 - 1$. That is, $d(v_2) = d(v_3) = d(v_7) = d(v_1)$. Thus $\max\{M_{v_j}^A | j = 2, 3, 7\}$ is the perfect sequence of the tree. For v_7 we have that $RB(v_7) = \{v_1, v_2, \dots, v_7\} \cup \{v_{10}, v_{11}, v_{12}\}$ and $\langle Bd'(v_8) \rangle = \langle 0 \rangle = \langle Bd'(v_9) \rangle$. Thus $T' = \{RB'(v_7), E(RB'(v_7))\}$ is the subtree produced by the root subbranch of v_7 . (See Figure 3-7.)

Since $v_7 \in A(v_2)$, we get the v_7 -vertex sequence set

$$S_{v_7}(T') = \{ \begin{aligned} &\langle v_2, v_1, v_{10}, v_{11}, v_{12}, v_3, v_4, v_5, v_6 \rangle, \\ &\langle v_2, v_1, v_{10}, v_{11}, v_{12}, v_3, v_6, v_4, v_5 \rangle, \\ &\langle v_2, v_1, v_{12}, v_{10}, v_{11}, v_3, v_4, v_5, v_6 \rangle, \\ &\langle v_2, v_1, v_{12}, v_{10}, v_{11}, v_3, v_6, v_4, v_5 \rangle, \\ &\langle v_2, v_3, v_4, v_5, v_6, v_1, v_{10}, v_{11}, v_{12} \rangle, \\ &\langle v_2, v_3, v_4, v_5, v_6, v_1, v_{12}, v_{10}, v_{11} \rangle, \\ &\langle v_2, v_3, v_6, v_4, v_5, v_1, v_{10}, v_{11}, v_{12} \rangle, \\ &\langle v_2, v_3, v_6, v_4, v_5, v_1, v_{12}, v_{10}, v_{11} \rangle \}. \end{aligned}$$

Then the v_7 -degree sequence set of $S_{v_7}(T')$ is

$$\begin{aligned} D_{v_7}(T') &= \{ \begin{aligned} &\langle 2, 2, 1, 0, 0, 2, 1, 0, 0 \rangle, \\ &\langle 2, 2, 1, 0, 0, 2, 0, 1, 0 \rangle, \\ &\langle 2, 2, 0, 1, 0, 2, 1, 0, 0 \rangle, \\ &\langle 2, 2, 0, 1, 0, 2, 0, 1, 0 \rangle, \\ &\langle 2, 2, 1, 0, 0, 2, 1, 0, 0 \rangle, \\ &\langle 2, 2, 1, 0, 0, 2, 0, 1, 0 \rangle, \\ &\langle 2, 2, 0, 1, 0, 2, 1, 0, 0 \rangle, \\ &\langle 2, 2, 0, 1, 0, 2, 0, 1, 0 \rangle \} \\ &= \{ \begin{aligned} &\langle 2, 2, 1, 0, 0, 2, 1, 0, 0 \rangle, \\ &\langle 2, 2, 1, 0, 0, 2, 0, 1, 0 \rangle, \\ &\langle 2, 2, 0, 1, 0, 2, 1, 0, 0 \rangle, \\ &\langle 2, 2, 0, 1, 0, 2, 0, 1, 0 \rangle \}. \end{aligned} \end{aligned}$$

Therefore we have the maximum sequence $mdR(v_7)$ in $D_{v_7}(T')$ as $mdR(v_7) = \langle 2, 2, 1, 0, 0, 2, 1, 0, 0 \rangle$. Since $mdR(v_7) = \langle 2, 2, 1, 0, 0, 2, 1, 0, 0 \rangle > \langle 0 \rangle = \langle Bd'(v_8) \rangle = \langle Bd'(v_9) \rangle$, we have $M_{v_7}^A = \langle 3, 2, 2, 1, 0, 0, 2, 1, 0, 0, 0 \rangle$.

In the same way, we get

$$\begin{aligned} M_{v_2}^A &= \langle 3, 2, 1, 0, 0, 2, 1, 0, 0, 2, 0, 0 \rangle \\ M_{v_3}^A &= \langle 3, 2, 2, 1, 0, 0, 2, 0, 0, 1, 0, 0 \rangle. \end{aligned}$$

Therefore $\max\{M_{v_j} | j = 2, 3, 7\} = M_{v_7}$. Since

$$\begin{aligned} M_{v_7}^A &= \langle 3, 2, 2, 1, 0, 0, 2, 1, 0, 0, 0, 0 \rangle \\ &> \langle 3, 2, 2, 1, 0, 0, 2, 0, 0, 1, 0, 0 \rangle \\ &= A, \end{aligned}$$

$M_{v_7}^A$, which is B in the Example 3.6.6, is the perfect sequence of the tree.

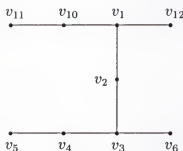


Figure 3-7

By the Tree Classification Theorem (Theorem 3.5.5) and the previous example, we have the following important corollary.

Corollary 3.6.9 *Let $A = \langle d_1, d_2, \dots, d_n \rangle$ be a nonnegative integer sequence satisfying the DSC and the OSbC. If $d_1 - 1 > d_i$ for every $i = 2, 3, \dots, n$, then A is the perfect sequence for an n -tree.*

We will see in the following example how we construct a tree shape from a given maximal degree sequence.

Example 3.6.10 Let $D = \langle d_1, d_2, \dots, d_{10} \rangle = \langle 3, 2, 1, 2, 0, 0, 1, 0, 0, 0 \rangle$ be a maximal degree sequence. Then we expect a 10-tree, T . Let $\hat{s} = \langle v_1, v_2, \dots, v_{10} \rangle$ be the vertex sequence having D as the degree sequence. Then $d(v_1) = d_1 = 3$, so the vertex v_1

has 3 edges. (See Figure 3-8.) Though there is no specific order for these edges, we label them as e_1, e_2, e_3 for the convenience of explanation.

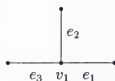


Figure 3-8

From the definition of vertex sequence (Definition 3.3.4) v_2 is one of the elements of the adjacent set of v_1 , $A(v_1)$. Without loss of generality, let $e_1 = \{v_1, v_2\}$. Then v_2 has 2 'new' edges, e_4 and e_5 , since $d_2 = 2$. Let $e_4 = \{v_2, v_3\}$. By the same reasoning, v_3 has 1 'new' edge, named e_6 . So $e_6 = \{v_3, v_4\}$. Since $d_4 = 2$, v_4 has 2 'new' edges, named e_7 and e_8 . (See Figure 3-9.)

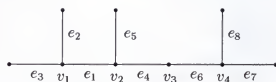


Figure 3-9

Since $d_5 = 0$ and $d_6 = 0$, neither v_5 nor v_6 has a 'new' edge; that is, v_5 and v_6 are end vertices. On the other hand, v_5 and v_6 are the adjacent vertices of v_4 , since $d_4 = 2$. Thus v_7 is an adjacent vertex of one of $\{v_1, v_2, v_3\}$. Since $d_3 = 1$ and v_4 is the adjacent vertex of v_3 , v_3 is not an adjacent vertex of v_7 . Since $d_2 = 2$ and we have only one adjacent vertex of v_2 , namely v_3 , we add an adjacent vertex, v_7 . (See Figure 3-10.)

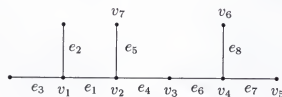


Figure 3-10

Since $d_7 = 1$, v_7 has 1 'new' edge which connects with v_8 . Since $d_8 = 0$, there is no 'new' edge from v_8 . Since we covered up to d_2 , the remaining two vertices, v_9 and v_{10} , are adjacent vertices of v_1 which has two 'unused' edges, e_2 and e_3 . Let $e_2 = \{v_1, v_9\}$, and $e_3 = \{v_1, v_{10}\}$. Since we are thinking about an unlabeled tree and we put the edge names for our convenience, there are no specific names for the vertices and edges. Thus we get the following 10-tree. (See Figure 3-11.)

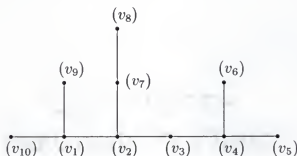


Figure 3-11

3.7 Extended Partitions and the Algorithm

In Definition 3.5.3 and the following explanation, it was shown that any tree can be represented by a unique perfect sequence. So, in this section, we develop an algorithm to produce all the perfect sequences for the n -trees for any positive integer n . From these perfect sequences, we get the shape of each n -tree as shown in the previous example, as well as the total number of n -trees.

From Definition 3.4.1 and the First Reduced Degree Sum Formula (Theorem 3.4.6), we see that a degree sequence for an $(n+1)$ -tree always has $n+1$ nonnegative integers as entries and these integers always add up to n which is the number of the edges of the given $(n+1)$ -tree. Thus, our first step in listing all possible degree sequences is to form ordered partitions of the integer n as sums of smaller positive integers.

Definition 3.7.1 An *ordered partition* of the positive integer n is a sequence of positive integers, $\langle p_1, p_2, \dots, p_m \rangle$, such that $\sum_{i=1}^m p_i = n$.

Note that $1 \leq m \leq n$ in the definition above.

Definition 3.7.2 An *extended partition* of an ordered partition $\langle p_1, p_2, \dots, p_m \rangle$ of the positive integer n is a sequence of $n+1$ nonnegative integers, $\langle d_1, d_2, \dots, d_{n+1} \rangle$, such that

- (i) $d_1 = p_1$, $\sum_{i=1}^{n+1} d_i = n$, and
- (ii) there is a subsequence $\langle d_{j_1}, d_{j_2}, \dots, d_{j_m} \rangle$ such that $d_{j_k} = p_k$ for $k = 1, \dots, m$.

Note that any entry d_i in the extended partition must be 0 unless it is a member of the ordered partition. It is important that different extended partitions are produced not only by the different ordered partitions but can also be produced by the same ordered partition.

For example, let $\langle d_1, d_2, \dots, d_{n+1} \rangle$ and $\langle d'_1, d'_2, \dots, d'_{n+1} \rangle$ be two extended partitions of an ordered partition $\langle p_1, p_2, \dots, p_m \rangle$, and let $\langle d_{j_1}, d_{j_2}, \dots, d_{j_m} \rangle$ and $\langle d'_{t_1}, d'_{t_2}, \dots, d'_{t_m} \rangle$ be subsequences, respectively, such that $d_{j_1} = d_1 = p_1 = d'_{t_1}$ and $d_{j_k} = p_k = d'_{t_k}$ for $k = 2, 3, \dots, m$. Assume that $j_i \neq t_i$ for at least one i . Then there are sections of zeros with different lengths between $d_{j_{(k-1)}}$ and d_{j_k} , and $d'_{t_{(k-1)}}$ and d'_{t_k} , respectively. So the extended partitions are different. Thus the order is very important in an extended partition.

Also note that the length of the extended partition of n must be $n+1$. So, for any given ordered partition, $\langle p_1, p_2, \dots, p_m \rangle$, we need to add $n-m+1$ zeros to make an extended partition.

Proposition 3.7.3 Let $\langle d_1, d_2, \dots, d_{n+1} \rangle$ be an extended partition of an ordered partition $P = \langle p_1, p_2, \dots, p_m \rangle$ of the integer n . Then this extended partition satisfies the First Reduced Degree Sum Formula (Theorem 3.4.6).

Proof From the given extended partition $\langle d_1, d_2, \dots, d_{n+1} \rangle$ of an ordered partition $\langle d_{j_1}, d_{j_2}, \dots, d_{j_m} \rangle$ of the integer n , we have that $\sum_{i=1}^{n+1} d_i = \sum_{i=1}^m d_{j_i} = n$ and

$$\begin{aligned} d_1 + \sum_{i=2}^{n+1} (d_i - 1) &= d_1 + \sum_{k=2}^m (d_{j_k} - 1) + \sum_{i=1, d_i \notin P}^{n+1} (d_i - 1) \quad \text{since } d_1 \neq 0 \\ &= d_{j_1} + \sum_{k=2}^m (d_{j_k} - 1) + \sum_{i=1, d_i \notin P}^{n+1} (d_i - 1) \\ &= \sum_{k=1}^m d_{j_k} + \sum_{k=2}^m (-1) + \sum_{i=1, d_i \notin P}^{n+1} (-1) \\ &= n + (m-1)(-1) + (n-m+1)(-1), \\ &= n - m + 1 - n + m - 1 = 0. \perp \end{aligned}$$

Even though we have a very large number of extended partitions of n , some of them can not be a maximal degree sequence for an $(n+1)$ -tree. So we first eliminate some ordered partitions that could not possibly result in maximal degree sequences. To do so, we need to check some properties of perfect sequences.

Proposition 3.7.4 *Let $D_s = \langle d_1, d_2, \dots, d_{n+1} \rangle$ be the perfect sequence of a given $(n+1)$ -tree T . If $d_i \neq 0$ and $d_{i+1} = 0$, then $0 = d_{i+1} = d_{i+2} = \dots = d_{i+d_i}$.*

Proof Since $d_i \neq 0$, the vertex v_i has d_i 'new' edges; that is, there are d_i adjacent vertices of v_i and they are labeled by d_i consecutive integers after i . Since D_s is a perfect sequence, it satisfies the OSbC. That is, the vertex v_{i+1} is a vertex of the largest degree among the adjacent vertices of v_i . But $d_{i+1} = 0$. Therefore we have $d_j = 0$ for $j = i+2, i+3, \dots, i+d_i$. $\perp$

This proposition gives us a method to make an extended partition from an ordered partition by adding zeros. That is, we know certain extended partitions cannot be perfect sequences just by checking this property.

Proposition 3.7.5 *Let $D_s = \langle d_1, d_2, \dots, d_{n+1} \rangle$ be the perfect sequence of a given $(n+1)$ -tree T . If $n \geq 2$, then $d_1 \geq 2$.*

Proof If $d_1 = 1$, then $d_i = 0$ for every $i = 2, 3, \dots, n+1$ by Proposition 3.6.2. Since $n \geq 2$, there are at least 3 vertices in T . Thus we have that $d_1 + \sum_{i=2}^{n+1} (d_i - 1) = 1 + \sum_{i=2}^{n+1} (-1) \leq 1 + (-1 - 1) = -1 \neq 0$. This is a contradiction to the First Reduced Degree Sum Formula. Therefore $d_1 \geq 2$. $\perp$

According to the above proposition, the only trees whose perfect sequences start with 0 or 1 are the tree with zero edge and the tree with one edge. The only degree sequence representation of a tree with zero edge is $\langle 0 \rangle$, and the only degree sequence representation of a tree with one edge is $\langle 1, 0 \rangle$. These are easy enough to visualize and count, so for the algorithm we work on the assumption that we are listing trees with at least two edges. Then the first term of the perfect sequence will be at least 2.

The following proposition shows that we have an arc shape tree if the first term of the perfect sequence is 2.

Proposition 3.7.6 *If $D_s = \langle d_1, d_2, \dots, d_{n+1} \rangle$ is the perfect sequence of an $(n+1)$ -tree T and $d_1 = 2$, then $d_i = 1$ for $i = 2, 3, \dots, n-1$ and $d_n = d_{n+1} = 0$.*

Proof By Proposition 3.6.2 we have that $d_i \in \{0, 1\}$ for any $i = 2, 3, \dots, n+1$, since $d_1 = 2$. Then the possible highest degree in T is degree 2. So T is just a linear tree with n edges. That is, there are exactly two end vertices which cannot be v_1 . Therefore there are exactly two zeros in the sequence. Write a degree sequence for T , in the order of the vertices in the following figure.



Figure 3-12

That is, start at a vertex adjacent to an end vertex and move in the longer direction first. Clearly the degree sequence resulting from this method contains the two zeros for v_n and v_{n+1} . Then we will get a degree sequence of the form $D = \langle 2, 1, 1, \dots, 1, 0, 0 \rangle$. For the same tree, the numbers in a degree sequence cannot change, but only the order in which they occur can change by changing the position of zeros. And, clearly, the new ordered sequence $< D$. So the sequence D must be the perfect sequence of T . That is, $D_s = D$. $\perp$

Most of the extended partitions of n start with a 1 or a 2. Since the last two propositions classify trees whose degree sequences start with these values, we do not need to consider ordered partitions that start with 1 or 2.

Proposition 3.7.6 shows that for any given number n of edges there is only one tree with n edges whose perfect sequence starts with a 2. In particular, it is a linear tree with n edges, since $d_i = 1$ for $i = 2, 3, \dots, n-1$. We add this sequence at the end of the algorithm.

By Proposition 3.6.2, we do not need the ordered partitions in which the largest summand appears more than once or is not the first term in the ordered partition.

If an ordered partition has either one or two entries only, then we can produce only one extended partition for each case. See the following Proposition 3.7.7 and Corollary 3.7.8.

Proposition 3.7.7 *Let $D_s = \langle d_1, d_2, \dots, d_{n+1} \rangle$ be the perfect sequence of an $(n+1)$ -tree T with $d_2 = 0$. Then $d_1 = n$ and $d_i = 0$ for all $i > 1$.*

Proof: Let $\hat{s}$ be a vertex sequence yielding a perfect sequence D_s of an $(n+1)$ -tree T . Since D_s is a perfect sequence and $0 = d_2 = d(v_2) - 1$, all of the vertices in the tree other than v_1 have degree 1, so all the terms in D_s other than d_1 must be zero. By the First Reduced Degree Sum Formula (Theorem 3.4.6), we have $d_1 = n$. $\perp$

Corollary 3.7.8 *If an ordered partition $P = \langle p_1, p_2 \rangle$ of n has only two entries, then the produced extended partition, $\langle d_1, d_2, \dots, d_{n+1} \rangle$, from P is such that $d_1 = p_1$, $d_2 = p_2$ and $d_i = 0$ for $i = 3, 4, \dots, n+1$.*

We demonstrated several necessary conditions which apply to a perfect sequence of a tree. Unfortunately, while these conditions are sufficient in almost all cases, there are certain sequences which, although they satisfy all the conditions, are not perfect sequences of the trees they represent. The problems usually arise when the tree has two or more vertices of the largest degree. We saw a counterexample in Example 3.6.6.

Note that in the Example 3.6.6 the two degree sequences A and B have the same ordered partition, $P = \langle 3, 2, 2, 1, 2, 1 \rangle$. That is, our algorithm produces the degree sequence B as one of the extended partitions of the ordered partition P . Therefore all we need to do is just drop the degree sequence A from the result of our algorithm.

Each step described so far has involved checking each member of a list of possible degree sequences and removing those which could not be the perfect sequence of a tree. The last step is no exception. Given a degree sequence, we check if there is more than one vertex of maximal degree. If there are more than one (in Example 3.6.6, both vertices v_1 and $v_7 = u_1$ have the same degree, 3), we write degree sequences for the tree starting at each such vertex. If the largest of these with respect to the ordering $\leq$ is the sequence we are checking, then it is a perfect sequence. If some other vertex gives a larger degree sequence, it is false that the sequence we are checking is a perfect sequence. This is a surefire approach in that, given a degree sequence, it checks for a perfect sequence by considering every possible alternative for the maximum and seeing if the sequence in hand is the largest.

CHAPTER 4 TREES AS INVARIANTS FOR BRAIN TUMOR SHAPES

4.1 Introduction

A sphere packing treatment plan for a given brain tumor is produced in the first chapter. This plan can be represented numerically as a list of parameters of the dose spheres. In particular, the list consists of the coordinates of the dose center in millimeters along the lateral (Lat), anterior-posterior (AP), and axial (Ax) axes together with the radius of the dose sphere in millimeters. From these data, we can assign a unique corresponding graph by matching a sphere with a vertex and matching the adjacency of two spheres with an edge. Then, by using the notion of *cutvertex*, we give an order to the vertex set. We use this order to decide which edges to choose in order to obtain a unique *maximal tree* contained in the graph.

Our main theorem of this section is Theorem 4.5.6: *The perfect sequences are invariants of the shapes of arbitrary brain tumors.* We then develop a *Mathematica* program (see Appendix B) to get a maximal tree of the graph for a sphere packing plan for a given brain tumor. In the near future, we plan to use this program to classify all the treatment plans which have been done at the Brain Institute of the University of Florida. We hope that the classification gives some insight about brain tumor shapes.

In this chapter, we assume that a brain tumor is connected, so the graph representation for any brain tumor is a connected graph.

4.2 Definitions and Some Properties

Definition 4.2.1 A graph $G = (V(G), E(G))$ consists of a vertex set $V(G) = \{v_1, \dots, v_n\}$ and an edge set $E(G) = \{e_1, \dots, e_m\} \subset \{\{u, v\} | u, v \in V(G)\}$ in which each edge is an unordered pair of two distinct vertices u and v , and the edge $\{u, v\}$ is said to join u and v . For an edge $\{u, v\}$ we say that u and v are *adjacent vertices* and the vertex u and the edge $\{u, v\}$ are *incident* with each other, as are v and $\{u, v\}$. From now on, $|V(G)|$ denotes the number of vertices and $|E(G)|$ denotes the number of edges. For a vertex $u \in V(G)$, the vertex set

$$A_G(u) = \{v \in G | v \text{ is an adjacent vertex of } u\}$$

is called the *adjacent set* of u in G .

Since there is no loop represented by a single edge, in a graph in this article, no vertex is adjacent to itself.

Definition 4.2.2 We say that a graph G is *isomorphic* to a graph H if and only if there exists a bijection $f : V(G) \rightarrow V(H)$ such that $\{u, v\} \in E(G)$ if and only if $\{f(u), f(v)\} \in E(H)$.

Definition 4.2.3 A *walk* is an alternating sequence $\langle v_0, e_1, v_1, e_2, \dots, e_k, v_k \rangle$ of vertices and edges s.t. $v_{i-1} \neq v_i$ for $i = 1, 2, \dots, k$, and $e_i = \{v_{i-1}, v_i\}$ for every i . A *cycle* is a walk $\langle v_0, e_1, v_1, e_2, \dots, e_k, v_k \rangle$ with $k \geq 3$ in which $v_0 = v_k$ is the only vertex repeated. A *path* is a walk with no repeated vertex. A (u, v) -*path* is a path with the first vertex u and the last vertex v . A (u, u) -*path* is a vertex u . The *length* of the (u, v) -path, $l(u, v)$, is the number of edges in the (u, v) -path. So the length of (u, u) -path is 0. The *distance* $d(u, v)$ between two vertices u and v is the length of a shortest path joining them, if any; otherwise $d(u, v) = \infty$.

Note that a path of length ≥ 1 is of linear shape.

Definition 4.2.4 Let $G = (V, E)$ be a graph where $V = \{v_1, v_2, \dots, v_n\}$. Then, for any vertex $v \in V$, we say that $P_G(v) = j$ if and only if

$j = \min\{i \in \{1, 2, \dots, n\} \mid \text{there is a } (v, v_i)\text{-path in } G\}.$

We call $P_G(v)$ the *path-value*, or simply the *P-value* of the vertex v in G .

Definition 4.2.5 Let $G = (V, E)$ be a graph. For any nonempty subset $V' \subset V$, let $F = \{ \{u, v\} \in E \mid u, v \in V' \} \subset E$. For any subset F' of F , the graph $G' = (V', F')$ is called a *subgraph* of G . A graph G is *connected* if and only if every pair of vertices are joined by a path. A graph having no cycle is *acyclic*. Note that an acyclic graph can be disconnected. A *forest* is an acyclic graph. A *tree* is a connected acyclic graph. An *n-tree* is a tree with n vertices.

Definition 4.2.6 The *degree* of a vertex v , denoted by $d(v)$, is the sum of the number of edges incident with v .

Definition 4.2.7 A *cutvertex* of a graph is a vertex of degree greater than 2, whose removal increases the number of components of the graph.

Definition 4.2.8 A *nonseparable* graph is connected, nontrivial, and has no cutvertex. A *block* of a graph is a maximal nonseparable subgraph.

Thus if v is a cutvertex of a connected graph G , then $G - v$ is disconnected. Note that the traditional definition of cutvertex (usually called *cutpoint* in graph theory) does not have the degree condition. But we do not want to separate any linear shape subgraph into several blocks because there is no difficulty in handling linear shape subgraphs in our algorithm. Thus we do not call a vertex with degree 2, a cutvertex in this chapter, even though it separates the graph.

And if H is a block, then none of the vertices in H is a cutvertex of H . Note that if there is no cutvertex in G , then the graph G itself is a block. This definition differs from the definition of block in Harary [5], because we have a different definition of cutvertex. Because of this, we have Proposition 4.2.10 below.

Proposition 4.2.9 *If a graph G is of linear shape, then the maximal tree of G is G itself.*

Proposition 4.2.10 [5] *Let v be a vertex of a connected graph G . The following statements are equivalent:*

- (1) v separates G and $d(v) \geq 2$.
- (2) There exist vertices u and w distinct from v such that v is on every (u, w) -path.
- (3) There exists a partition of the set of vertices $V(G) - \{v\}$ into subsets U and W such that for any vertices $u \in U$ and $w \in W$, the vertex v is on every (u, w) -path.

Definition 4.2.11 Let $G = \{V, E\}$ be a graph. For any vertex $v \in V$ we define the edge subset $I_G(v) \subset E(G)$ by $I_G(v) = \{\{v, u\} \in E(G) \mid u \in A_G(v)\}$. We call $I_G(v)$ the *incident edge set* of v in G .

Clearly, $I_G(v)$ contains all the incident edges of the vertex v in G .

Definition 4.2.12 Let $G = \{V, E\}$ be a graph. For any subset F of E we call the set $V(F) = \{v \in V(G) \mid \text{there is an edge } e \in F \text{ such that } v \text{ and } e \text{ are incident}\}$ the *F vertex set*.

Proposition 4.2.13 [5] *The following statements are equivalent for a graph T :*

- (1) T is a tree.
- (2) Every two vertices of T are joined by a unique path.
- (3) T is connected and $|V(T)| = |E(T)| + 1$.
- (4) T is acyclic and $|V(T)| = |E(T)| + 1$.

Note that if a tree has n vertices, then there are exactly $n - 1$ edges.

Definition 4.2.14 Let $G = \{V, E\}$ be a connected graph and E' a subset of E . A connected subgraph $T = \{V, E'\}$ is called a *maximal tree* of G if and only if $T' = \{V, E' \cup \{e\}\}$ contains a cycle for any edge $e \in E - E'$.

Our work in this chapter produces a particular maximal tree for any given graph. So we have the following (well-known) theorem.

Theorem 4.2.15 *Every connected graph has a maximal tree.*

Example 4.2.16 Let G be a graph with 14 vertices and 18 edges as in the figure 4-1 below.

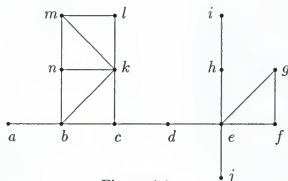


Figure 4-1

There are 3 cutvertices, b, c, e . So the given graph G can be separated into 6 blocks as follows:

$$\begin{aligned}
 G_1 &= \{V_1, E_1\} = \{ \{a, b\}, \{ \{a, b\} \} \}, \\
 G_2 &= \{V_2, E_2\} = \{ \{b, c, k, l, m, n\}, \\
 &\quad \{ \{b, c\}, \{c, k\}, \{k, l\}, \{l, m\}, \{m, n\}, \{n, b\}, \{b, k\}, \{k, m\} \} \}, \\
 G_3 &= \{V_3, E_3\} = \{ \{c, d, e\}, \{ \{c, d\}, \{d, e\} \} \}, \\
 G_4 &= \{V_4, E_4\} = \{ \{e, f, g\}, \{ \{e, f\}, \{f, g\}, \{g, e\} \} \}, \\
 G_5 &= \{V_5, E_5\} = \{ \{e, h, i\}, \{ \{e, h\}, \{h, i\} \} \}, \text{ and} \\
 G_6 &= \{V_6, E_6\} = \{ \{e, j\}, \{ \{e, j\} \} \}.
 \end{aligned}$$

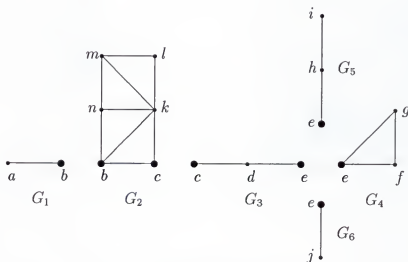


Figure 4-2

4.3 Graphs for Sphere Packing Plans

The sphere packing treatment plan S having n spheres for a given brain tumor, from Chapter 1, consists of the centers, $c_1, c_2, \dots, c_n$, and the radii, $r_1, r_2, \dots, r_n$, of the spheres with $c_i = (x_i, y_i, z_i)$ for every i , where x_i is for the AP coordinate, y_i is for the Lat coordinate and z_i is for the Ax coordinate of the center, c_i . (Note that the physicians use diameters, rather than radii, in their data sets.)

We consider the spheres to be the vertices of a graph, and we assign the vertex, v_i , for the sphere centered at c_i with radius r_i . Now we can assign an edge between two vertices if the spheres assigned by these two vertices meet. In Chapter 1, we noted that the collimators do not deliver clear cut boundaries for spheres—there is some spillover. The neurosurgeons currently assume that the portion between two spheres gets enough dosage of radiation if the distance between the centers of the two spheres is less than or equal to eleven tenths of the sum of the radii of the two spheres. Thus for any two vertices assigned for two spheres satisfying this condition we assign an edge between them. Then we have the following proposition.

Proposition 4.3.1 *Let $G = \{V, E\}$ be a graph for the sphere packing plan of a brain tumor. For any vertices v_i and v_j in V , $d(c_i, c_j) \leq \frac{11}{10}(r_i + r_j)$ if and only if $\{v_i, v_j\} \in E$. Thus we have the following edge set;*

$$E = \{\{i, j\} | d(c_i, c_j) \leq (11/10)(r_i + r_j) \text{ where } i < j, i = 1, 2, \dots, n-1\}.$$

Since any brain tumor is connected in this article, there is at least one (u, v) -path in G for every two vertices, u and v in G . Thus we have the following proposition.

Proposition 4.3.2 *The graph $G = \{V, E\}$ produced by the above method for a given tumor is a connected graph.*

Since we are dealing with unlabeled graphs in this article, we have the following proposition.

Proposition 4.3.3 *For any sphere packing treatment plan for a given brain tumor, there exists a unique graph that we assign to the treatment plan.*

By means of the above method, we get a unique connected graph $G = \{V, E\}$ for the sphere packing treatment plan for the tumor. This graph does not have the information about the positions of the dose spheres as an aspect of the structure and looks only at how the spheres are attached. It is clear that the graph G does not have a loop or multiple edges between two vertices, since every edge is assigned for two different vertices which are assigned for two attached spheres.

4.4 Order in the Vertex Set

To get a maximal tree from a graph we may need to delete some edges in the graph. Thus we need to label each vertex in order to choose certain edges to delete, even though we are dealing with an unlabeled graph in this chapter. So there is no specific meaning for this labeling except that it is only used to choose a maximal tree for the graph.

Example 4.4.1 Let P be the sphere packing treatment plan (see Figure 4-3) for a given brain tumor.



Figure 4-3

Then, by Proposition 4.3.3, there is a unique graph G for the sphere packing plan. The graph is given below in Figure 4-4.



Figure 4-4

There are two different isomorphism classes, namely T_1 and T_2 , of maximal trees for the graph G . That is, we could choose either of these for a maximal tree to assign the graph G . Since, in our algorithm, we want to get the same maximal tree for a given graph every time, we need certain rules to get the same tree. (See Figure 4-5.)



Figure 4-5

In this chapter, the graph comes from a sphere packing treatment plan. That is, the main shape of the brain tumor depends on the size of spheres and the connection between spheres. Even though we cannot keep the information about the size of spheres in the graph, it seems to us that the bigger the sphere is, the more it has a chance to get attached to more spheres. So we assume that the larger degree vertices play a more important role in classifying the tumor shapes into trees than the smaller degree vertices. That is, by choosing a largest degree vertex first, the shape of the tumor is apparently most closely preserved. Thus, in the previous example, we choose the vertex q or s as the starting vertex; then we pick all the incident edges. Then we get the tree T_1 . Therefore we want to choose the tree T_1 for the maximal tree of G . Note that the tree T_2 produces a linear graph which does not show the shape of the tumor as closely as T_1 .

But there are some vertices which are more important than the bigger degree vertices. Recall the graph G in Example 4.2.16. Then, in the block G_2 , it is clear that the vertex k has the largest degree, 5. But the cutvertices, b and c , play a critical role in obtaining a maximal tree of G . So, to get a maximal tree for a graph G , we want to start at the cutvertices of G first, instead of the vertices with the largest degree. So we want to label the cutvertices first. On the other hand, there

are different kinds of 'cutvertices' in some graphs. The following example shows such a case.

Example 4.4.2 Let $G = \{V, E\}$ be the graph in Figure 4-6.

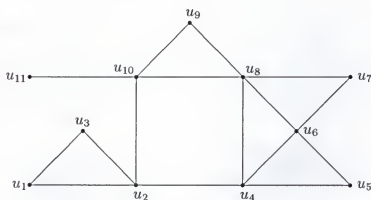


Figure 4-6

Then there are two cutvertices of the graph G , namely u_2 and u_{10} . Using these two cutvertices, the graph G is separated into four subgraphs, namely G_1 , G_2 , G_3 and G_4 . (See Figure 4-7.)

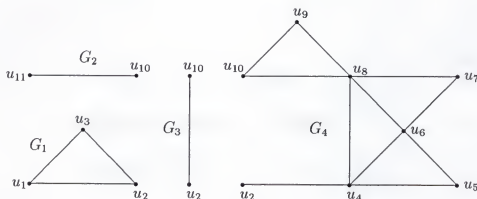


Figure 4-7

Then the subgraphs G_1 , G_2 , and G_3 are blocks, but the subgraph G_4 contains its own cutvertices, u_4 and u_8 , which are not cutvertices of the graph G . We separate the subgraph G_4 into four subgraphs by using its cutvertices u_4 and u_8 .

(See Figure 4-8.) Note that there is no specific order in labeling the subgraphs. So at this moment, we relabel the subgraphs of G by $H_1, H_2, \dots, H_7$.

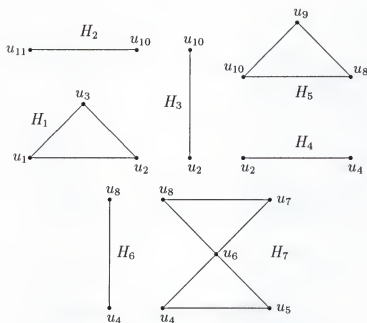


Figure 4-8

Then, again, the subgraphs $H_1, H_2, \dots, H_6$ are blocks, but the subgraph H_7 contains its own cutvertex, u_6 , which is neither a cutvertex of the graph G nor a cutvertex of the subgraph G_4 . We separate this subgraph H_7 into two subgraphs by using u_6 . (See Figure 4-9.) And we relabel the subgraphs of G by $B_1, B_2, \dots, B_8$.

Therefore $\{u_2, u_{10}\}$ is the set of cutvertices of the graph G . But $\{u_4, u_8\}$ is the set of the cutvertices of a subgraph which is produced after the first separation using the cutvertices of the graph, and u_6 is the cutvertex of a subgraph which is produced after the second separation.

For the purpose of labeling the vertices in a graph, we separate these cutvertex sets for different levels, from each other.

Definition 4.4.3 Let G be the graph assigned for a given sphere packing plan P . Then the vertex set $V(G)$ of the graph G is the union of two disjoint subsets, namely $C_1(G)$ and $C_1^c(G)$, where $C_1(G) = \{v \in V(G) \mid v \text{ is a cutvertex of } G\}$

and $C_1^c(G) = V(G) - C_1(G)$. We call $C_1(G)$ the *first step cutvertex set* of G . For every $i = 2, 3, \dots, |V(G)|$, we define a subset $C_i(G)$ of $C_{i-1}^c(G)$ as the collection of the vertices of $C_{i-1}^c(G)$, which are cutvertices of a subgraph produced after the $(i-1)$ -th separation using the elements of $C_{i-1}(G)$. Then we call $C_i(G)$ the *i-th step cutvertex set* of G , and let $C_i^c(G) = C_{i-1}^c(G) - C_i(G)$. We assume that there are finitely many vertices in a given graph. If there exists at least one cutvertex, then there exists an integer $1 \leq k \leq n$ such that $C_i(G) \neq \emptyset$ for every $i \leq k$, and $C_i(G) = \emptyset$ for every $i \geq k+1$. Then we call k the *separation step constant* of the graph G , and let $C^c(G) = (C_k)^c(G)$.

If there is no cutvertex of G then the graph is a block, and we say that the separation step constant of G is 0.

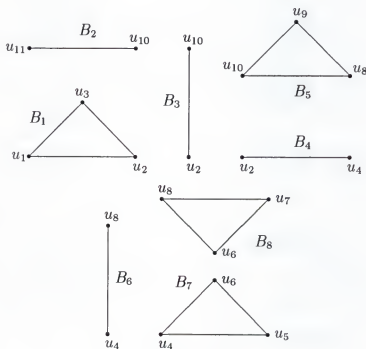


Figure 4-9.

Proposition 4.4.4 Let $|V(G)| = n$ and let the separation step constant of G be $k > 0$. Then there is a sequence of integers $\{m_1, m_2, \dots, m_k\}$ with $0 < m_1 < m_2 < \dots < m_k$ such that $|C_1(G)| = m_1$, and $|C_i(G)| = m_i - m_{i-1}$, for $i = 2, 3, \dots, k$.

Proof Since $k > 0$, the first step cutvertex set $C_1(G)$ is not empty. Thus there is a positive integer, namely m_1 , such that $|C_1(G)| = m_1$. For every $i = 2, 3, \dots, k$, let $m_i = m_{i-1} + |C_i(G)|$. For every $i = 2, 3, \dots, k$, the i -th step cutvertex set $C_i(G)$ is not empty. So we have that $m_i > m_{i-1}$, and $|C_i(G)| = m_i - m_{i-1}$. $\perp$

Now we label the vertices as follows;

- (1) the vertices in $C_1(G)$ are labeled by $\{1, 2, \dots, m_1\}$,
- (2) for every $i = 2, 3, \dots, k$, the vertices in $C_i(G)$ are labeled by $\{m_{i-1} + 1, m_{i-1} + 2, \dots, m_i - 1, m_i\}$, and
- (3) the vertices in $C^c(G)$ are labeled by $\{m_k + 1, m_k + 2, \dots, n\}$.

We arbitrarily name the vertices of $C_i(G)$ and $C^c(G)$, and denote that $C_1(G) = \{u_1, u_2, \dots, u_{m_1}\}$, $C_i(G) = \{u_{m_{i-1}+1}, u_{m_{i-1}+2}, \dots, u_{m_i}\}$ for $i = 2, 3, \dots, k$, and $C^c(G) = \{u_{m_k+1}, u_{m_k+2}, \dots, u_n\}$. It is clear that for every vertex $u_p \in C_i(G)$ and every vertex $u_q \in C_j(G)$ with $i \neq j$, we have that $p < q$ if and only if $i < j$. And also, for every vertex $u_p \in C_i(G)$ for every $i = 1, 2, \dots, k$, and every vertex $u_q \in C^c(G)$, we have that $p < q$. Assume that, for every vertex u_i , the center and the radius of the sphere which is assigned to that vertex u_i , are $c_i = \{x_i, y_i, z_i\}$ and r_i , respectively.

We now relabel the elements for each of the sets, $C_1(G), C_2(G), \dots, C_k(G)$ and $C^c(G)$, in the following way:

for every pair of vertices, u_i, u_j , in one of $C_1(G), C_2(G), \dots, C_k(G)$ and $C^c(G)$, $i < j$ if and only if

- (i) $d(u_i) > d(u_j)$; or
- (ii) If $d(u_i) = d(u_j)$, then $r_i > r_j$; or
- (iii) If $d(u_i) = d(u_j)$ and $r_i = r_j$, then $x_i < x_j$; or
- (iii) If $d(u_i) = d(u_j)$, $r_i = r_j$, and $x_i = x_j$, then $y_i < y_j$; or
- (iv) If $d(u_i) = d(u_j)$, $r_i = r_j$, $x_i = x_j$, and $y_i = y_j$, then $z_i < z_j$.

Note that $d(u_i)$ represents degree of the vertex u_i in G .

It is clear that for any two different spheres, at least one of the coordinates is different. Thus any two distinct spheres can always be compared by the above order. Therefore the above order for the vertices of the graph is well defined.

4.5 Maximal Tree from a Graph

In this section, we develop an algorithm that will uniquely select a maximal tree from the graph which is produced for a given brain tumor. The choice is made so that the tree will most closely resemble, in our opinion, the shape of the tumor - that is, of the graph obtained for the tumor.

Let $G = (V, E)$ be the graph for a given sphere packing plan. Let $V(G) = \{u_1, u_2, \dots, u_n\}$ for some positive integer n , where all the vertices in V are labeled by the method in the previous section.

Define a subgraph $T_0 = (V(G), F_0)$ with $V(T_0) = V = V(G)$ and $F_0 = \emptyset$; that is, there is no edge in T_0 . Then, for every $i = 1, 2, \dots, n$, we have that $P_{T_0}(u_i) = i$. (See Definition 4.2.4.)

(1) For the vertex u_1 , let $F_1 = F_0 \cup I_G(u_1)$. (See Definition 4.2.11.) Let $T_1 = (V(G), F_1)$. The subgraph $(V(F_1), F_1) \subset T_1$ is a tree. Thus, if $V(F_1) = V(G)$, then $T_1 = (V(G), F_1) = (V(F_1), F_1)$ is our maximal tree of G . Note that for every vertex $v \in V(F_1)$ we have that $P_{T_1}(v) = 1$, and for every vertex $u_j \notin V(F_1)$, we have that $P_{T_1}(u_j) = j$.

(2) Assume that $V(F_1) \neq V(G)$. For the vertex u_2 , there are two cases, that is, $P_{T_1}(u_2) = 1$ and $P_{T_1}(u_2) = 2$. Now define the edge subset $I'_G(u_2)$ by $I'_G(u_2) = I_G(u_2) - \{\{u_2, v\} \in E \mid P_{T_1}(v) = 1\}$.

(a) If $P_{T_1}(u_2) = 1$, that is, there is a (u_1, u_2) -path in the subgraph T_1 , then let $F_2 = F_1 \cup I'_G(u_2)$. Each edge in the set $\{\{u_2, v\} \in E \mid P_{T_1}(v) = 1\} = \{\{u_2, v\} \in E \mid v \in V(F_1)\}$ makes a cycle with $\{u_1, u_2\}$ and $\{u_1, v\}$, since $P_{T_1}(u_2) = 1$. Thus there is no cycle in the subgraph $(V(F_2), F_2)$, so it is a tree. Let $T_2 = (V(G), F_2) \subset G$.

If $V(F_2) = V(G)$, then $T_2 = (V(G), F_2) = (V(F_2), F_2)$ is our maximal tree of G . Note that for every vertex $v \in V(F_2)$ we have that $P_{T_2}(v) = 1$, and for every vertex $u_j \notin V(F_2)$ we have that $P_{T_2}(u_j) = j$.

(b) If $P_{T_1}(u_2) = 2$, then there is no (u_1, u_2) -path in the subgraph T_1 .

(i) If there is a non-empty subset $\{u_{2_1}, u_{2_2}, \dots, u_{2_t}\} \subset V(I_G(u_2))$ where $P_{T_1}(u_{2_i}) = 1$ for $i = 1, 2, \dots, t$ and $2_i < 2_j$ for $1 \leq i < j \leq t$, then let $F_2 = F_1 \cup I'_G(u_2) \cup \{\{u_2, u_{2_1}\}\}$. Then the subgraph $(V(F_2), F_2) \subset G$ is a tree. Let $T_2 = (V(G), F_2) \subset G$. If $V(F_2) = V(G)$, then $T_2 = (V(G), F_2) = (V(F_2), F_2)$ is our maximal tree of G . Note that for every vertex $v \in V(F_2)$ we have that $P_{T_2}(v) = 1$, and for every vertex $u_j \notin V(F_2)$ we have that $P_{T_2}(u_j) = j$.

(ii) If none of the vertices in $V(I_G(u_2))$ has p-value 1, that is, for every vertex $v \in V(I_G(u_2))$ there is no (u_1, v) -path, then let $F_2 = F_1 \cup I_G(u_2)$. Since $P_{T_1}(w) = 1$ for every vertex $w \in F_1$ and $P_{T_1}(v) = 2$ for every vertex $v \in I_G(u_2)$, the subgraph $(V(F_2), F_2)$ is a forest, but not a tree. Let $T_2 = (V(G), F_2) \subset G$. It is clear that the subgraph T_2 cannot be our maximal tree, even if we have that $V(F_2) = V(G)$, because the subgraph T_2 is disconnected. Note that for every vertex $v \in V(F_2) = V(F_1 \cup I_G(u_2)) = V(F_1) \cup V(I_G(u_2))$ we have that $P_{T_2}(v) = 1$ if $v \in V(F_1)$, and $P_{T_2}(v) = 2$ if $v \in V(I_G(u_2))$. And, for every vertex $u_j \notin V(F_2)$, we have that $P_{T_2}(u_j) = j$.

(3) By induction, at the j -th step, the subgraph $T_{j-1} = (V(G), F_{j-1})$ has been defined. We want to define the subgraph $T_j \subset G$. For the vertex u_j , there are two cases, that is, $P_{T_{j-1}}(u_j) = j$ and $P_{T_{j-1}}(u_j) = q$ for some $1 \leq q < j$. For every $i = 1, 2, \dots, n$, define

$$m_{j-1}(i) = \min\{l \in \{1, 2, \dots, n\} | P_{T_{j-1}}(u_l) = i \text{ and } u_l \in A_G(u_j)\}.$$

If the set $\{l \in \{1, 2, \dots, n\} | u_l \in V(I_G(u_j)) \text{ and } P_{T_{j-1}}(u_l) = i\}$ is empty, then the vertex $u_{m_{j-1}(i)}$ is not defined. So there is no edge $\{u_j, u_{m_{j-1}(i)}\}$.

(a) Let $P_{T_{j-1}}(u_j) = j$; that is, for every $i = 1, 2, \dots, j-1, j+1, \dots, n$, there is no (u_j, u_i) -path in the subgraph $T_{j-1} = (V(G), F_{j-1})$. Let

$$F_j = F_{j-1} \cup \left(\bigcup_{i=1}^n \{ \{u_j, u_{m_{j-1}(i)}\} \} \right).$$

Then the subgraph $(V(F_j), F_j)$ is a forest. Let $T_j = (V(G), F_j) \subset G$. If $V(F_j) = V(G)$ and $|V(F_j)| = |F_j| + 1$, then, by Proposition 4.2.13, the subgraph $T_j = (V(G), F_j) = (V(F_j), F_j)$ is our maximal tree of G .

(b) Let $P_{T_{j-1}}(u_j) = q$ for some $1 \leq q < j$, that is, there is a (u_q, u_j) -path in the subgraph $T_{j-1} = (V(G), F_{j-1})$. Now let

$$F_j = F_{j-1} \cup \left(\bigcup_{i=1, i \neq q}^n \{ \{u_j, u_{m_{j-1}(i)}\} \} \right).$$

Then the subgraph $(V(F_j), F_j)$ is a forest. Let $T_j = (V(G), F_j) \subset G$. If $V(F_j) = V(G)$ and $|V(F_j)| = |F_j| + 1$, then by Proposition 4.2.13, the subgraph $T_j = (V(G), F_j) = (V(F_j), F_j)$ is our maximal tree of G .

Note that if u is an i -th step cutvertex, then $d(u)$ in G is same as $d(u)$ in T_{i-1} . Also note that, for every $i = 1, 2, \dots, n$, $\{u_1, u_2, \dots, u_i\} \subset V(F_i)$, that is, $V(G) = V(F_n)$. We will see in Theorem 4.5.3 that $T_n = (V(G), F_n) = (V(F_n), F_n)$ is our maximal tree of G . By the Tree Classification Theorem (Theorem 3.6.8), this tree T_n can be denoted by a unique perfect sequence.

Example 4.5.1 Let $G = (V, E)$ be a graph where $V = \{u_1, u_2, \dots, u_8\}$, which are ordered as in the previous subsection, and $E = \{e_1, e_2, \dots, e_{13}\}$ as in the figure below. Clearly, the graph G does not have a cutvertex, so the order is decided by only the sizes of the spheres and the coordinates of the centers of the spheres.

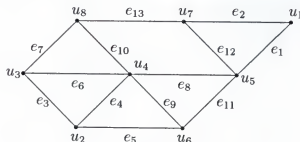


Figure 4-10

Let $T_0 = (V(G), F_0)$ be a subgraph of $G = (V, E)$ where $F_0 = \emptyset$, as shown in Figure 4-11.

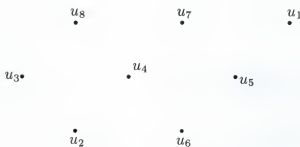


Figure 4-11

Since $I_G(u_1) = \{e_1, e_2\}$, let $F_1 = F_0 \cup I_G(u_1) = \{e_1, e_2\}$. Then $V(F_1) = \{u_1, u_5, u_7\}$. Since $V(F_1) = \{u_1, u_5, u_7\} \neq V(G)$, the tree $(V(F_1), F_1)$ is not a maximal tree of G . Let $T_1 = (V(G), F_1)$ which is shown in Figure 4-12.

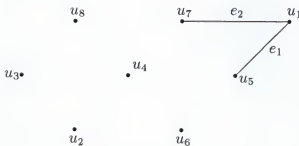


Figure 4-12

We have that $m_1(i) = i$ for $i = 1, 2, 3, 4, 6$, and 8 , but $m_1(5)$ and $m_1(7)$ do not exist. Since $P_{T_1}(u_2) = 2$ and $A_G(u_2) = \{u_3, u_4, u_6\}$, let

$$\begin{aligned} F_2 &= F_1 \cup (\cup_{i=3,4,6} \{u_2, u_i\}) \\ &= F_1 \cup I_G(u_2) \\ &= \{e_1, e_2\} \cup \{e_3, e_4, e_5\} = \{e_1, e_2, e_3, e_4, e_5\}. \end{aligned}$$

Then $V(F_2) = \{u_1, u_5, u_7, u_2, u_3, u_4, u_6\} \neq V(G)$ and the subgraph $(V(F_2), F_2)$ is a forest, not a tree. Thus it cannot be a maximal tree of G . Let $T_2 = (V(G), F_2)$. (See Figure 4-13.)

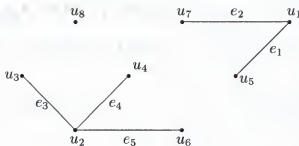


Figure 4-13

Note that $P_{T_2}(u_1) = P_{T_2}(u_5) = P_{T_2}(u_7) = 1$, $P_{T_2}(u_2) = P_{T_2}(u_3) = P_{T_2}(u_4) = P_{T_2}(u_6) = 2$, and $P_{T_2}(u_8) = 8$. Thus we have that $m_2(1) = 1, m_2(2) = 2, m_2(8) = 8$. Note that $m_2(i)$ does not exist for $i = 3, 4, 5, 6, 7$. Since $u_1 \notin A(u_3)$ and $P_{T_2}(u_3) = 2$, we get $F_3 = F_2 \cup \{u_3, u_8\} = \{e_1, e_2, e_3, e_4, e_5, e_7\}$. Since the subgraph $(V(F_3), F_3)$ is a forest, but not a tree, it is not a maximal tree of G . Define $T_3 = (V(G), F_3)$. (See Figure 4-14.)

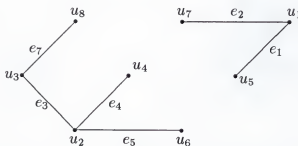


Figure 4-14

Note that $P_{T_3}(u_1) = P_{T_3}(u_5) = P_{T_3}(u_7) = 1$ and $P_{T_3}(u_2) = P_{T_3}(u_3) = P_{T_3}(u_4) = P_{T_3}(u_6) = P_{T_3}(u_8) = 2$. Thus we have that $m_3(1) = 5$ and $m_3(2) = 4$. Note that $m_3(i)$ does not exist for $i = 3, 4, \dots, 8$. Since $P_{T_3}(u_4) = 2$, we define $F_4 = F_3 \cup \{u_4, u_5\} = \{e_1, e_2, e_3, e_4, e_5, e_7\}$. Since $V(F_4) = \{v_1, v_2, \dots, v_8\} = V(G)$ and $|V(F_4)| = 8 = |F_4| + 1$, the subgraph $T = (V(F_4), F_4)$ is a maximal tree of G . (See Figure 4-15.)

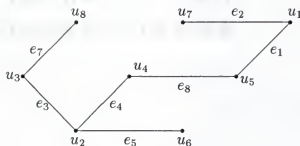


Figure 4-15

At this moment, we have that $P_T(u_1) = P_T(u_2) = \dots = P_T(u_8) = 1$. Then, for every $u_t \in V(G)$ with $t > 4$, we have that $F_t = F_4$. Thus, for every $t > 4$, the subgraph $(V(F_t), F_t)$ is same as $T = (V(F_4), F_4)$.

The next example shows how we get the maximal tree for a given graph containing some cutvertices.

Example 4.5.2 Let G be the graph in Example 4.2.16. (See Figure 4-1.) Then $C_1(G) = \{b, c, e\}$ and $C_1^c(G) = \{a, d, f, g, h, i, j, k, l, m, n\}$. In $C_1(G)$, we have that $d(e) = 5$, $d(b) = 4$ and $d(c) = 3$. On the other hand, in $C_1^c(G)$, we have that $d(k) = 5$, $d(m) = d(n) = 3$, $d(d) = d(f) = d(g) = d(h) = d(l) = 2$, $d(a) = d(i) = d(j) = 1$. And $C_2 = \emptyset$. Thus we order the vertices as follows; $v_1 = e$, $v_2 = b$, $v_3 = c$ and $v_4 = k$, $\{v_5, v_6\} = \{m, n\}$, $\{v_7, v_8, v_9, v_{10}, v_{11}\} = \{d, f, g, h, l\}$, $\{v_{12}, v_{13}, v_{14}\} = \{a, i, j\}$. Assume that $v_5 = m$, $v_6 = n$, $v_7 = d$, $v_8 = f$, $v_9 = g$, $v_{10} = h$, $v_{11} = l$, $v_{12} = a$, $v_{13} = i$, $v_{14} = j$, which are decided by the sizes and centers of the corresponding spheres. (See Figure 4-16.)

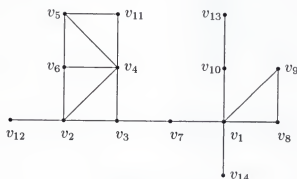


Figure 4-16

Thus it cannot be a maximal tree of G . We have the following subgraph which is $T_3 = (V(G), F_3)$. (See Figure 4-19.)

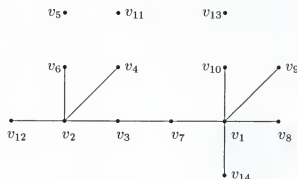


Figure 4-19

Since $A_G(v_4) = \{v_2, v_3, v_5, v_6, v_{11}\}$, and

$$P_{T_3}(v_2) = P_{T_3}(v_3) = P_{T_3}(v_6) = 1 = P_{T_3}(v_4), P_{T_3}(v_5) = 5, P_{T_3}(v_{11}) = 11,$$

we define $F_4 = F_3 \cup \{\{v_4, v_5\}\} \cup \{\{v_4, v_{11}\}\}$. Then the subgraph $(V(F_4), F_4)$ is a tree. But $V(F_4) \neq V(G)$. Thus it cannot be a maximal tree of G . Then, for the cutvertex v_4 , we have that the following subgraph which is $T_4 = (V(G), F_4)$. (See Figure 4-20.)

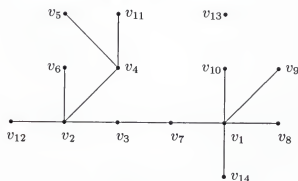


Figure 4-20

For the vertices v_5, v_6, v_7, v_8, v_9 , there is no 'new' edge. Thus we get the same subgraph as above, that is, $T_4 = T_5 = T_6 = T_7 = T_8 = T_9$. But, for the cutvertex v_{10} , we have that $v_{13} \in V(I_G(v_{10}))$ and $P_{T_3}(v_{13}) = 13 \neq 1 = P_{T_3}(v_{10})$. Thus

$F_{10} = F_9 \cup \{v_{10}, v_{13}\}$ and $V(F_{10}) = V(G)$. Therefore we have the following subgraph which is $T = (V(G), F_{10}) = (V(F_{10}), F_{10})$. (See Figure 4-21.)

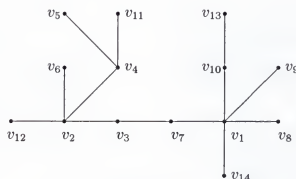


Figure 4-21

Clearly, the subgraph $T = (V(F_{10}), F_{10})$ does not include a cycle. Thus it is a maximal tree of G . That is, for every $t = 11, 12, 13, 14$, we have that $(V(F_t), F_t) = (V(F_{10}), F_{10})$. Therefore, the subgraph $T = (V(F_{10}), F_{10}) = (V(G), F_{10})$ is the maximal tree of the graph G .

For this resulting tree $T = (V(F_{10}), F_{10})$, choose a vertex sequence $\hat{s}$ by $\hat{s} = \langle v_1, v_7, v_3, v_2, v_4, v_5, v_{11}, v_6, v_{12}, v_{10}, v_{13}, v_8, v_9, v_{14} \rangle$. Then the sequence

$$P = \langle 5, 1, 1, 3, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0 \rangle$$

is the degree sequence of $\hat{s}$. By Theorem 3.4.12, P satisfies DSC. It is clear that P also satisfies DSbC. By Corollary 3.6.10, P is the perfect sequence for the tree $T = (V(F_{10}), F_{10})$.

Theorem 4.5.3 *Let $G = \{V, E\}$ be a graph for a sphere packing plan where $V = \{u_1, u_2, \dots, u_n\}$ with the order described in Section 4.4. Let the subgraph $(V(F_n), F_n)$ of G come from the above procedure. Then $V(F_n) = V(G)$. Therefore $(V(F_n), F_n)$ is a maximal tree of G and there is a unique perfect n -sequence for the tree $(V(F_n), F_n)$.*

Proof The subgraph $(V(F_n), F_n)$ is connected. In fact, assume that the subgraph $(V(F_n), F_n)$ contains two disjoint subtrees, namely T_1 and T_2 . Then the

path-value for every vertex in T_1 is different from the path-value for every vertex in T_2 . Since the graph G is connected, there is at least one edge $e \in E(G)$ which connects the two subgraphs T_1 and T_2 and $e \notin E(T_1 \cup T_2) = F_n$. Let $\{ \{v_{j_1}, v_{k_1}\}, \{v_{j_2}, v_{k_2}\}, \dots, \{v_{j_q}, v_{k_q}\} \}$ be the collection of such edges where $v_{j_i} \in T_1$ and $v_{k_i} \in T_2$ for $i = 1, 2, \dots, q$. Without loss of generality, let $j_1 < j_i$ for every $i = 2, 3, \dots, q$. Then it is clear that $\{v_{j_1}, v_{k_1}\} \in F_{j_1}$ by the procedure of getting the subgraph $(V(F_n), F_n)$. Then $\{v_{j_1}, v_{k_1}\} \in F_n$. This is a contradiction. Therefore the subgraph $(V(F_n), F_n)$ is connected. The path-value in the procedure of getting the subgraph $(V(F_n), F_n)$ guarantees that the subgraph $(V(F_n), F_n)$ is acyclic. Therefore $(V(F_n), F_n)$ is a tree. It is clear that, for every $i = 1, 2, \dots, n$, the vertex set $V(F_i)$ contains the vertices $u_1, u_2, \dots, u_i$. Thus $V(F_n)$ contains all the vertices of G , since $V(F_n) \supset \{u_1, u_2, \dots, u_n\} = V(G)$. That is, $V(F_n) = V = V(G)$. Hence the subgraph $(V(F_n), F_n)$ is a maximal tree of G . By the Tree Classification Theorem (Theorem 3.6.8), it is clear that there is a perfect n -sequence for the tree $(V(F_n), F_n)$. $\perp$

Definition 4.5.4 The *tree shape* of a brain tumor is the tree corresponding to that tumor, or equivalently, its unique corresponding perfect sequence.

Definition 4.5.5 Two tumors have the *same shape* if and only if their corresponding graphs obtained from their respective sphere packing plans are the same.

As in Chapter 2, for any brain tumor, we have a unique sphere packing plan which can be denoted by a graph. From the previous theorem, we have a unique perfect sequence for the graph. That is, we can assign exactly one perfect sequence to each brain tumor. So, if two perfect sequences are distinct, then their corresponding trees, and therefore their respective corresponding graphs and sphere packings, are also distinct. That is, if two tumors are represented by distinct perfect sequences, then their corresponding trees are not isomorphic. And their respective graphs and

sphere packing plans are not isomorphic. Thus, we may consider their shapes to be distinct. Therefore we have the following as our main theorem of this section.

Theorem 4.5.6 *The perfect sequences are invariants of the shapes of arbitrary brain tumors.*

4.6 Graph from a Maximal Tree

Let $T = \{V, E'\}$ be a maximal tree for a graph $G = \{V, E\}$ which is obtained from the sphere packing plan of a given tumor or solid shape. We observe that we can recover the original graph, G , from T . Since $V(T) = V = V(G)$ and $E(T) = E' \subset E = E(G)$, we need to add the remaining edges to the edge set of the maximal tree to get the graph G . Since the graph G comes from the sphere packing treatment plan, each vertex v in G denotes a sphere of the sphere packing plan. Thus, for every vertex $v_i \in G$, let $c_i = (x_i, y_i, z_i)$ and r_i be the center and the radius of the sphere. By Proposition 4.3.1, for any two vertices v_i and v_j in V if $d(c_i, c_j) \leq \frac{11}{10}(r_i + r_j)$, then the edge $\{v_i, v_j\}$ is in the edge set, $E(G)$, of the graph G . Thus

$$E' \cup \{\{v_i, v_j\} \in V \times V \mid d(c_i, c_j) \leq \frac{11}{10}(r_i + r_j)\} = E.$$

Therefore we get the graph $G = \{V, E\}$.

4.7 Algorithm for Our Maximal Tree

Let $G = (V, E)$ be the graph for a sphere packing plan with $|V| = n$.

STEP A Find all the i -th step cutvertices of the graph G , for $1 \leq i \leq n$.

STEP B Label the vertices in the vertex set V by the order explained in Section 4.4.

STEP C By induction, for each $i = 1, 2, \dots, n$, get the subgraph $(V(F_i), F_i)$ of G . Then $T = (V(F_n), F_n)$ is our maximal tree.

APPENDIX A

SHELLING PROGRAM FOR SPHERE PACKING

In this appendix, we provide the C++ program we developed to get the sphere packing plan for a given tumor shape. The first section shows the program translating the data from CT scan to the data of appropriate form for the main program, which is in the second section. Note that this program does not include the *score function* which was developed by Thomas Wagner. We refer the reader Appendix E for the complete article on the sphere packing treatment plan.

A.1 Program for Data Reading

```
// rewrite data file into x,y,z format
//   & produce the total # of data, x,y,z min & max

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <math.h>

main()
{ const int nofdata=3370;           // comes from data003.txt
  int xposition[nofdata], yposition[nofdata], zposition[nofdata];
  int q;
  float a; float b; float c;
  int k=0;

  { ifstream inClientFile("data003.txt", ios::in);
    while (inClientFile >> a >> b >> c)
    { xposition[k] = int(floor(a)); // check the number
      yposition[k] = int(floor(b)); // check the number
      zposition[k] = int(floor(c)); // check the number
      k++;
    }
  }

  int xmin=xposition[0];
  int xmax=xposition[0];
  int ymin=yposition[0];
  int ymax=yposition[0];
  int zmin=zposition[0];
  int zmax=zposition[0];

  for (int i=0; i<nofdata; i++)
  { if ( xposition[i] < xmin )
      xmin = xposition[i];
```

```

        else if ( xposition[i] > xmax )
            xmax = xposition[i];
        else if ( yposition[i] < ymin )
            ymin = yposition[i];
        else if ( yposition[i] > ymax )
            ymax = yposition[i];
        else if ( zposition[i] < zmin )
            zmin = zposition[i];
        else if ( zposition[i] > zmax )
            zmax = zposition[i];
    }

    // output the result of centers
    { ofstream outClientFile("mM003.txt", ios::app);
      { outClientFile << nofdata << setw(10) << xmin << setw(5)
        << xmax << setw(5) << ymin << setw(5) << ymax
        << setw(5) << zmin << setw(5) << zmax << endl;
      }
    }

    cout << nofdata << setw(5) << xmin << setw(5) << xmax
         << setw(5) << ymin << setw(5) << ymax << setw(5)
         << zmin << setw(5) << zmax;
    cin >> q;
}

```

A.2 Program for Center Searching

// searching the centers for pat 2

```

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <math.h>

int square(int);           // function prototype
int MAX(int,int,int);      // function prototype

main()
{ const int nofdata=3370;  // check the number from mM003.txt
  const int xmin = -32;    // check the number
  const int xmax = -4;     // check the number
  const int ymin = -29;    // check the number
  const int ymax = 4;      // check the number
  const int zmin = -57;    // check the number
  const int zmax = -37;    // check the number

  const int xfix = 1-xmin;
  const int yfix = 1-ymin;
  const int zfix = 1-zmin;

  int xposition[nofdata], yposition[nofdata], zposition[nofdata];

```

```

int q;
float a; float b; float c;
int k=0;

{ ifstream inClientFile("data003.txt", ios::in);
  while (inClientFile >> a >> b >> c)
  { xposition[k] = int(floor(a))+xfix;
    yposition[k] = int(floor(b))+yfix;
    zposition[k] = int(floor(c))+zfix;
    k++;
  }
}

const int x = xmax - xmin + 3;
const int y = ymax - ymin + 3;
const int z = zmax - zmin + 3;
const int nc = x*y*z; // number of cells
int cc[nc]; // status value

for (int i=0; i<nc; ++i)
  cc[i] = -3;

int xc[nc]; // x-coord. of c[nc]
int yc[nc]; // y-coord. of c[nc]
int zc[nc]; // z-coord. of c[nc]

// setting the x,y,z-coord. for each cell
for (int i=0; i<nc; i++)
{ zc[i] = floor(i/(y*x));
  xc[i] = floor((i-zc[i]*y*x)/x);
  yc[i] = i-zc[i]*y*x-xc[i]*x;
}

// setting the boundary of tumor by -1
for (int i=0; i<nofdata; i++)
{ cc[zposition[i]*x*y + xposition[i]*y + yposition[i]] = -1;
}

// setting the initial of normal by 0
cc[0] = 0;

// setting the normal cell by 0
int h=0;
for (int j=0; j<x*y; j++)
{ for (int i=0; i<nc; i++)
  { if ( cc[i] == -3 )
    { if (cc[i-1]==0||cc[i+1]==0||cc[i-y]==0||cc[i+y]==0)
      { cc[i] = 0;
        h++;
      }
    }
    else;
  }
  else;
}
}

```

```

    if (h==0)
        break;
    else
        h=0;
}

// setting the remaining tumor cell by -2
for (int i=0; i<nc; i++) // checking
{ if ( cc[i] == -3 ) // checking
    cc[i] = -2; // checking
} // checking

// setting the boundary cells
int maxrad;
int nofmaxrad;
for (int j=0; j<=MAX(x,y,z)/2+1; j++)
{ int k=0;
  for (int i=0; i<nc; i++)
  { if (cc[i] == -1 || cc[i] == -2)
    { k++;
      if ( cc[i-1] == j || cc[i+1] ==j || cc[i-y]==j
          || cc[i+y]==j || cc[i-y*x]==j || cc[i+y*x]==j )
      { cc[i] = j+1;
        nofmaxrad = k;
      }
      else;
    }
    else;
  }
  maxrad = j;
  if (k==0)
    break;
  else;
}

// checking the no. of nofmaxrad
// if nofmaxrad > 100, then you have change the number
if (nofmaxrad >100)
{ cout << endl << endl << endl << endl << endl <<
  << endl << endl << endl << endl;
  cin >> q;
}
else;

// pick the boxes with maxrad
int suprad[100];
int TotalNofBox4[100];
int TotalNofMax=0;
for (int i=0; i<nc; i++)
{ if (cc[i]==maxrad)

```

```

    { suprad[TotalNofMax]=i;
      TotalNofMax++;
    }
    else;
}

cout << TotalNofMax << endl;
cin >> q;

// Count for the number of centers of each case
for (int k=0; k<TotalNofMax; k++)
{ // reset TotalNofBox4[ ] to 0
  for (int i=0; i<100; i++)
  { TotalNofBox4[i] = 0;
  }
  TotalNofBox4[maxrad]++;

  int tcc[nc]; // temporary cell notion
  for (int i = 0; i<nc; ++i)
    tcc[i] = cc[i]; // reassign tcc[i] for 'for' loop

  // remove the biggest sphere
  for (int p=0; p<nc; ++p)
  { int dist = floor(sqrt( square(yc[suprad[k]]-yc[p])
    + square(xc[suprad[k]]-xc[p])
    + square(zc[suprad[k]]-zc[p]) ));
    if ( dist < maxrad )
      tcc[p] = 0;
    else;
  }

  // setting the remaining tumor cell by -2
  for (int i=0; i<nc; i++)
  { if ( tcc[i] >= 1 )
    { tcc[i] = -2;
    }
    else;
  }

  // setting the boundary cells
  for (int j=0; j<=MAX(x,y,z)/2+1; j++)
  { int l=0;
    for (int i=0; i<nc; i++)
    { if (tcc[i] == -2)
      { if ( tcc[i-1] == j || tcc[i+1] ==j || tcc[i-y]==j
        || tcc[i+y]==j || tcc[i-y*x]==j || tcc[i+y*x]==j )
        { tcc[i] = j+1;
          ++l;
        }
        else;
      }
      else;
    }
    if (l==0)
      break;
  }
}

```

```

    else;
}

// check if there is more maxrad pixel
for (int i=0; i<nc; i++)
{ if ( tcc[i] == maxrad )
  { cout << endl << endl << endl <<
    " WARNING!!! WE HAVE MORE 'MAXRAD' PIXEL AT ("
      << xc[i] << ", " << yc[i] << ", " << zc[i] << ")"
      << endl;
    cin >> q;
    break;
  }
  else;
}

//remove the smaller or = sphere
for (int j=maxrad-1; j>1; --j)
{ for (int i=0; i<nc; i++)
  { if ( tcc[i] ==j )
    { TotalNofBox4[j]++;
      // remove the sphere
      for (int l=0; l<nc; ++l)
      { int dist = floor(sqrt( square(yc[i]-yc[l])
                                + square(xc[i]-xc[l])
                                + square(zc[i]-zc[l]) ));

        if ( dist < j )
          tcc[l] = 0;
        else;
      }
      // setting the remaining tumor cell by -2
      for (int i=0; i<nc; i++)
      { if ( tcc[i] >= 1 )
        { tcc[i] = -2;
          else;
        }
        // setting the boundary cells
        for (int j=0; j<=MAX(x,y,z)/2+1; j++)
        { int l=0;
          for (int i=0; i<nc; i++)
          { if (tcc[i] == -2)
            { if ( tcc[i-1] == j || tcc[i+1] ==j ||
                  tcc[i-y]==j || tcc[i+y]==j ||
                  tcc[i-y*x]==j || tcc[i+y*x]==j )
              { tcc[i] = j+1;
                ++l;
              }
            }
            else;
          }
          else;
        }
      }
      if (l==0)
        break;
    }
  }
}

```

```

        else;
    }
}
else;
}
}

// output the result of centers
{ ofstream outClientFile("pat003-NofCenters.txt", ios::app);
  { outClientFile << endl << endl << endl << endl << endl <<
    (k+1) << setw(5) << "(" << (xc[suprad[k]]-xfix) << "," <<
    (yc[suprad[k]]-yfix) << "," << (zc[suprad[k]]-zfix)
    << ")" with radius = " << maxrad << endl;
  }
}

for (int j=maxrad; j>1; j--)
{ ofstream outClientFile("pat003-NofCenters.txt", ios::app);
  { outClientFile << setw(4) << j << setw(4)
    << TotalNofBox4[j] << endl;
  }
}

// Search for centers
for (int k=0; k<TotalNofMax; k++)
{ int tcc[nc]; // temporary cell notion
  for (int i = 0; i<nc; ++i)
    tcc[i] = cc[i]; // reassign tcc[i] for 'for' loop

  // output the result of centers
  { ofstream outClientFile("pat003-centers.txt", ios::app);
    { outClientFile << (k+1) << endl << setw(5) << "x"
      << setw(5) << "y" << setw(5) << "z" << setw(10)
      << "radius" << endl << setw(5) << (xc[suprad[k]]-xfix)
      << setw(5) << (yc[suprad[k]]-yfix) << setw(5)
      << (zc[suprad[k]]-zfix) << setw(10) << maxrad << endl;
    }
  }

  // remove the biggest sphere
  for (int p=0; p<nc; ++p)
  { int dist = floor(sqrt( square(yc[suprad[k]]-yc[p])
    + square(xc[suprad[k]]-xc[p])
    + square(zc[suprad[k]]-zc[p]) ));
    if ( dist < maxrad )
      tcc[p] = 0;
    else;
  }

  // setting the remaining tumor cell by -2
  for (int i=0; i<nc; i++)
  { if ( tcc[i] >= 1 )
    tcc[i] = -2;
  }
}

```

```

    else;
}

// setting the boundary cells
for (int j=0; j<=MAX(x,y,z)/2+1; j++)
{ int l=0;
  for (int i=0; i<nc; i++)
  { if (tcc[i] == -2)
    { if ( tcc[i-1] == j || tcc[i+1] ==j || tcc[i-y]==j
      || tcc[i+y]==j || tcc[i-y*x]==j || tcc[i+y*x]==j )
      { tcc[i] = j+1;
        ++l;
      }
    }
    else;
  }
  else;
}
if (l==0)
  break;
else;
}

// check if there is more maxrad pixel
for (int i=0; i<nc; i++)
{ if ( tcc[i] == maxrad )
  { cout << endl << endl << endl <<
    " WARNING!!! WE HAVE MORE 'MAXRAD' PIXEL AT ("
    << xc[i] << "," << yc[i] << "," << zc[i] << ")" << endl;
    cin >> q;
    break;
  }
  else;
}

//remove the smaller or = sphere
for (int j=maxrad-1; j>1; --j)
{ for (int i=0; i<nc; ++i)
  { if ( tcc[i] ==j )
    { TotalNofBox4[j]++;
      // output the result of centers
      { ofstream outClientFile("pat003-centers.txt",
                               ios::app);
        { outClientFile << setw(5) << (xc[i]-xfix)
          << setw(5) <<(yc[i]-yfix) << setw(5)
          << (zc[i]-zfix) << setw(10) << j << endl;
        }
      }
      // remove the sphere
      for (int l=0; l<nc; ++l)
      { int dist = floor(sqrt( square(yc[i]-yc[l])
        + square(xc[i]-xc[l])
        + square(zc[i]-zc[l]) ));
        if ( dist < j )
          tcc[l] = 0;
      }
    }
  }
}

```

```

        else;
    }
    // setting the remaining tumor cell by -2
    for (int i=0; i<nc; i++)
    { if ( tcc[i] >= 1 )
        tcc[i] = -2;
        else;
    }
    // setting the boundary cells
    for (int j=0; j<=MAX(x,y,z)/2+1; j++)
    { int l=0;
        for (int i=0; i<nc; i++)
        { if (tcc[i] == -2)
            { if ( tcc[i-1] == j || tcc[i+1] ==j ||
                tcc[i-y]==j || tcc[i+y]==j ||
                tcc[i-y*x]==j || tcc[i+y*x]==j )
                { tcc[i] = j+1;
                    ++l;
                }
            }
            else;
        }
        else;
    }
    if (l==0)
        break;
    else;
}
}
else;
}
}
{ ofstream outClientFile("pat003-centers.txt", ios::app);
  { outClientFile << endl;
  }
}
}
}
int square(int a)
{ return a*a;
}
int MAX(int a, int b, int c)
{ int p=0;
  p = a;
  if (a<=b)
    p = b;
  else if (a<=c)
    p = c;
  else;
  return p;
}

```

APPENDIX B

PROGRAM FOR THE MAXIMAL TREE OF A GRAPH

In this appendix, we provide the *Mathematica* program we developed in joint work with Matthew Harvey. The program below was written by Matthew Harvey to get the maximal tree for the graph which we assigned for a brain tumor. Note that this program does not use the method of ordering for the vertices as in Chapter 3. Instead this program uses some commands which are in the function package in the *Mathematica* program library. If anyone wants to develop a program like this using a different program language, we recommend the vertex ordering algorithm in Chapter 3.

```
<< DiscreteMath`Combinatorica`
Off[General::spell]

LexPrecedes[a_, b_] := Module[{i},
  For[i = 1, i <= Min[Length[a], Length[b]], i++,
    If[ a[[i]] != b[[i]],
      If[ a[[i]] > b[[i]], Return[True], Return[False] ] ]
  ];
  If[ Length[b] > Length[a], Return[False] ];
  Return[True]]

DegreePoint[a_, k_] := Count[Flatten[ToUnorderedPairs[a]], k];

IndSubgraph[Graph[g_, v_], s_List] := Module[{i, j, newmat, sc},
  newmat = g;
  sc = Complement[Range[Length[v]], s];
  For[i = 1, i <= Length[g], i++,
    For[j = 1, j <= Length[g], j++,
      If[MemberQ[sc, i] || MemberQ[sc, j], newmat[[i, j]] = 0]]];
  Graph[newmat, v]];

CutPoints[gr_] :=
  Select[ArticulationVertices[gr], DegreePoint[gr, #] > 2 &]

NonCutPartitions[gr_] := Module[{partitions, cpts, bccs, skeleton},
  partitions = {};
  bccs = Select[BiconnectedComponents[gr], Length[#] > 2 &];
  partitions = Union[partitions, bccs];
  skeleton =
    IndSubgraph[gr,
      Union[
        Complement[ Range[V[gr]], Flatten[bccs] ],
```

```

        CutPoints[gr]
    ]
];

partitions =
  Union[partitions,
    Select[ConnectedComponents[
      FromUnorderedPairs[
        Select[ToUnorderedPairs[
          skeleton], # !=
            Intersection[#,
              Flatten[
                Select[BiconnectedComponents[skeleton],
                  Length[#] > 2 &]
              ]
            ]
        ]
      ]
    ], Length[#] > 1
    &]
];

Table[
  Select[
    ToUnorderedPairs[gr],
    # == Intersection[#, partitions[[i]] ]
    &], {i, Length[partitions]}]]

```

APPENDIX C

PROGRAM FOR CLASSIFYING TREES BY SEQUENCES

In the first section, we provide the *Mathematica* program we developed in joint work with Matthew Harvey. This program was written by Matthew Harvey to get the maximal degree sequences for any given integer $n \geq 2$. And we show the program results up to $n = 10$ in the second section.

C.1 Program

```

Off[General::spell]
<< DiscreteMath`Combinatorica`

DoPartitions[n_] :=
Module[{i, k, templist, returnlist},
  If[n == 0, {},
    returnlist = {};
    For[k = n, k >= 1, k--,
      templist = DoPartitions[n - k];
      For[i = 1, i <= Dimensions[templist][[1]], i++,
        returnlist = Append[returnlist,
          Flatten[{k, templist[[i]]}]
        ]
      ];
    ]
  ];
  returnlist
];

Filter[thelist_] :=
Select[thelist,
  ! ( MemberQ[{1, 2}, #1[[1]]] || (Max[#1] != First[#1])
    || (Count[#1, Max[#1]] > 1)
  )
]&];

AddZerosToSeq[part_] :=
Module[{nZeros, n, i, j, k, sum, positions, seqs, withzeros},
  If[Length[part] == 1, Return[{Table[If[i == 1, part[[1]], 0],
    {i, part[[1]] + 1}]}]
];
  seqs = {};
  sum = Sum[part[[i]], {i, 1, Length[part]}];
  nZeros = (sum + 1) - Length[part];
  n = Table[k, {k, 3, sum}];
  positions = KSubsets[n, nZeros - 1];

```

```

For[i = 1, i <= Length[positions], i++,
  k = 2;
  withzeros = {First[part]};
  For[j = 2, j <= sum, j++,
    If[MemberQ[positions[[i]], j],
      withzeros = Append[withzeros, 0],
      withzeros = Append[withzeros, part[[k]]];
      k++
    ]
  ];
  withzeros = Append[withzeros, 0];
  seqs = Append[seqs, withzeros]
];
seqs
];

AddZeros[seq_] :=
Flatten[Table[ AddZerosToSeq[seq[[i]]], {i, 1, Length[seq]} ],1]

SubSeqs[dseq_] :=
Module[ {branchlist, branch, workseq, branches, i},
  branches = dseq[[1]];
  workseq = Drop[dseq, 1];
  branchlist = {};
  For[i = 1, i <= branches, i++,
    dsum = workseq[[1]];
    workseq = Drop[workseq, 1];
    branch = {dsum};
    While[dsum != 0, dsum += (workseq[[1]] - 1);
      branch = Append[ branch, workseq[[1]] ];
      workseq = Drop[workseq, 1]
    ];
    branchlist = Append[branchlist, branch]
  ];
  branchlist
]

LexPrecedes[a_, b_] :=
Module[{i},
  For[i = 1, i <= Min[Length[a], Length[b]], i++,
    If[a[[i]] != b[[i]],
      If[ a[[i]] > b[[i]], Return[True], Return[False]
    ]
  ];
  If[Length[b] > Length[a], Return[False]];
  Return[True]
]

WeightsQ[dseq_] :=
Module[{i, subseqs, flag},
  subseqs = SubSeqs[dseq];
  flag = 1;
  If[OrderedQ[subseqs, LexPrecedes],

```

```

    For[i = 1, i <= Length[subseqs], i++,
      If[ ! WeightsQ[subseqs[[i]]], flag=0, flag*=1 ]
    ],
    Return[False]
  ];
  If[ flag == 1, Return[True], Return[False] ]
]

```

```

DegreeSumQ[dseq_] :=
Module[{dsum},
  dsum = dseq[[1]];
  For[i = 2, i <= Length[dseq], i++,
    dsum += dseq[[i]] - 1;
    If[dsum == 0,
      If[i == Length[dseq], Return[True], Return[False] ]
    ]
  ];
  Return[False]
]

```

```

ExtractSubtree[dseq_] :=
Module[{rtotal},
  rtotal = dseq[[1]];
  i = 1;
  While[rtotal > 0, i++,
    rtotal += dseq[[i]] - 1;
  ];
  Take[dseq, i]
]

```

```

RewriteTree[dseq_, i_] :=
Module[{branchesoff, subtree, brlength, seq1st},
  If[i == 1, Return[dseq]];
  branchesoff = i - 1;
  While[ ( branchesoff > 1 && Sum[dseq[[k]] - 1,
    {k, branchesoff, i - 1}] < 0 ) ||
    dseq[[branchesoff]] == 0, branchesoff-- ];
  brlength = Length[ ExtractSubtree[Drop[dseq, i - 1]] ];
  subtree = Drop[dseq, {i, i + brlength - 1}];
  subtree[[branchesoff]]--;
  seq1st = SubSeqs[Take[dseq, {i, i + brlength - 1}]];
  seq1st = Append[seq1st, RewriteTree[subtree, branchesoff]];
  seq1st = Sort[seq1st, LexPrecedes[#1, #2] &];
  Flatten[{dseq[[i]] + 1, seq1st}]
]

```

```

RightOrderQ[dseq_] :=
Module[{pos, seqs, j, rewritten},
  pos = Flatten[ Position[dseq, dseq[[1]] - 1 ] ];
  If[pos == {}, Return[True]];
  seqs = {};
  For[j = 1, j <= Length[pos], j++,
    rewritten = RewriteTree[ dseq, pos[[j]] ];

```

```

        If[LexPrecedes[rewritten, dseq] && rewritten != dseq,
            Return[False]
        ]
    ];
    Return[True]
]

LemmasQ[dseq_] :=
DegreeSumQ[dseq] && WeightsQ[dseq] && RightOrderQ[dseq]

CheckLemmas[lstseq_] :=
Module[{retseq, twoseq, i, j, n, p},
    retseq = {};
    j = 0;
    p = 10;
    n = Length[lstseq];
    For[i = 1, i <= n, i++,
        If[ j > n/10,
            Print[p, "%"];
            p += 10;
            j = 0
        ];
        j++;
        If[LemmasQ[ lstseq[[i]] ],
            retseq = Append[ retseq, lstseq[[i]] ]
        ]
    ];
    twoseq = {2};
    If[Length[lstseq] > 0,
        For[i = 2, i < Length[ retseq[[1]] ] - 1, i++,
            twoseq = Append[twoseq, 1]
        ]
    ];
    twoseq = Join[twoseq, {0, 0}];
    retseq = Append[retseq, twoseq];
    retseq
]

DSTRecursive[dseq_, bvert_, hvert_, trparam_] :=
Module[{i, v, tree, tempseq},
    v = hvert;
    tree = trparam;
    If[dseq != {0}, tempseq = Drop[dseq, 1];
        For[i = 1, i <= dseq[[1]], i++,
            v++;
            tree = Append[tree, {bvert , v}];
            subseq = ExtractSubtree[tempseq];
            tempseq = Drop[tempseq, Length[subseq]];
            {tree, v} = DSTRecursive[subseq, v, v, tree];
        ];
    ];
    {tree, v}
];

```

```

DegreeSeqToTree[dseq_] :=
First[ DSTRecursive[dseq, 1, 1, {}] ];

ToGraphics[ckd_] :=
For[t = 1, t <= Length[ckd], t++,
  Print[ckd[[t]]];
  ShowGraph[RadialEmbedding[FromUnorderedPairs
    [ DegreeSeqToTree[ckd[[t]]] ]
  ]
]
]

```

C.2 Some Program Results

The notation for a degree sequence is $\langle d_1, d_2, \dots, d_n \rangle$. But the *Mathematica* program produces the sequence notation with { and }. So in this section, any set notation is actually denotes the degree sequence. Note that an n -tree has n vertices, so there are $n-1$ edges.

n=3 total : 1

{2, 0, 0}



n=4 total : 2

{3, 0, 0, 0},

{2, 1, 0, 0}



n=5 total : 3

{4, 0, 0, 0, 0},

{3, 1, 0, 0, 0},

{2, 1, 1, 0, 0}



n=6

total : 6

{5, 0, 0, 0, 0, 0}, {4, 1, 0, 0, 0, 0}, {3, 2, 0, 0, 0, 0},
 {3, 1, 0, 1, 0, 0}, {3, 1, 1, 0, 0, 0}, {2, 1, 1, 1, 0, 0}



n=7

total : 11

{6, 0, 0, 0, 0, 0, 0}, {5, 1, 0, 0, 0, 0, 0}, {4, 2, 0, 0, 0, 0, 0},
 {4, 1, 0, 1, 0, 0, 0}, {4, 1, 1, 0, 0, 0, 0}, {3, 2, 1, 0, 0, 0, 0},
 {3, 1, 2, 0, 0, 0, 0}, {3, 1, 0, 1, 0, 1, 0}, {3, 1, 1, 0, 1, 0, 0},
 {3, 1, 1, 1, 0, 0, 0}, {2, 1, 1, 1, 1, 0, 0}



n=8

total : 23

$\{7, 0, 0, 0, 0, 0, 0, 0\}$, $\{6, 1, 0, 0, 0, 0, 0, 0\}$,
 $\{5, 2, 0, 0, 0, 0, 0, 0\}$, $\{5, 1, 0, 1, 0, 0, 0, 0\}$,
 $\{5, 1, 1, 0, 0, 0, 0, 0\}$, $\{4, 3, 0, 0, 0, 0, 0, 0\}$,
 $\{4, 2, 0, 0, 1, 0, 0, 0\}$, $\{4, 2, 1, 0, 0, 0, 0, 0\}$,
 $\{4, 1, 2, 0, 0, 0, 0, 0\}$, $\{4, 1, 0, 1, 0, 1, 0, 0\}$,
 $\{4, 1, 1, 0, 1, 0, 0, 0\}$, $\{4, 1, 1, 1, 0, 0, 0, 0\}$,
 $\{3, 2, 2, 0, 0, 0, 0, 0\}$, $\{3, 2, 1, 0, 0, 1, 0, 0\}$,
 $\{3, 2, 1, 0, 1, 0, 0, 0\}$, $\{3, 2, 1, 1, 0, 0, 0, 0\}$,
 $\{3, 1, 2, 1, 0, 0, 0, 0\}$, $\{3, 1, 1, 2, 0, 0, 0, 0\}$,
 $\{3, 1, 1, 0, 1, 0, 1, 0\}$, $\{3, 1, 1, 0, 1, 1, 0, 0\}$,
 $\{3, 1, 1, 1, 0, 1, 0, 0\}$, $\{3, 1, 1, 1, 1, 0, 0, 0\}$,
 $\{2, 1, 1, 1, 1, 1, 0, 0\}$



n=9

total : 47

```

{8, 0, 0, 0, 0, 0, 0, 0, 0}, {7, 1, 0, 0, 0, 0, 0, 0, 0},
{6, 2, 0, 0, 0, 0, 0, 0, 0}, {6, 1, 0, 1, 0, 0, 0, 0, 0},
{6, 1, 1, 0, 0, 0, 0, 0, 0}, {5, 3, 0, 0, 0, 0, 0, 0, 0},
{5, 2, 0, 0, 1, 0, 0, 0, 0}, {5, 2, 1, 0, 0, 0, 0, 0, 0},
{5, 1, 2, 0, 0, 0, 0, 0, 0}, {5, 1, 0, 1, 0, 1, 0, 0, 0},
{5, 1, 1, 0, 1, 0, 0, 0, 0}, {5, 1, 1, 1, 0, 0, 0, 0, 0},
{4, 3, 1, 0, 0, 0, 0, 0, 0}, {4, 2, 0, 0, 2, 0, 0, 0, 0},
{4, 2, 2, 0, 0, 0, 0, 0, 0}, {4, 2, 0, 0, 1, 0, 1, 0, 0},
{4, 2, 0, 0, 1, 1, 0, 0, 0}, {4, 2, 1, 0, 0, 1, 0, 0, 0},
{4, 2, 1, 0, 1, 0, 0, 0, 0}, {4, 2, 1, 1, 0, 0, 0, 0, 0},
{4, 1, 3, 0, 0, 0, 0, 0, 0}, {4, 1, 2, 0, 0, 1, 0, 0, 0},
{4, 1, 2, 1, 0, 0, 0, 0, 0}, {4, 1, 1, 2, 0, 0, 0, 0, 0},
{4, 1, 0, 1, 0, 1, 0, 1, 0}, {4, 1, 1, 0, 1, 0, 1, 0, 0},
{4, 1, 1, 0, 1, 1, 0, 0, 0}, {4, 1, 1, 1, 0, 1, 0, 0, 0},
{4, 1, 1, 1, 1, 0, 0, 0, 0}, {3, 2, 2, 0, 0, 1, 0, 0, 0},
{3, 2, 2, 1, 0, 0, 0, 0, 0}, {3, 2, 1, 2, 0, 0, 0, 0, 0},
{3, 2, 1, 0, 1, 0, 1, 0, 0}, {3, 2, 1, 1, 0, 0, 1, 0, 0},
{3, 2, 1, 1, 0, 1, 0, 0, 0}, {3, 2, 1, 1, 1, 0, 0, 0, 0},
{3, 1, 2, 1, 0, 0, 1, 0, 0}, {3, 1, 2, 1, 0, 1, 0, 0, 0},
{3, 1, 2, 1, 1, 0, 0, 0, 0}, {3, 1, 1, 2, 1, 0, 0, 0, 0},
{3, 1, 1, 1, 2, 0, 0, 0, 0}, {3, 1, 1, 0, 1, 1, 0, 1, 0},
{3, 1, 1, 1, 0, 1, 0, 1, 0}, {3, 1, 1, 1, 0, 1, 1, 0, 0},
{3, 1, 1, 1, 1, 0, 1, 0, 0}, {3, 1, 1, 1, 1, 1, 0, 0, 0},
{2, 1, 1, 1, 1, 1, 1, 0, 0}

```

n=10

total : 106

```

{9, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {8, 1, 0, 0, 0, 0, 0, 0, 0, 0},
{7, 2, 0, 0, 0, 0, 0, 0, 0, 0}, {7, 1, 0, 1, 0, 0, 0, 0, 0, 0},
{7, 1, 1, 0, 0, 0, 0, 0, 0, 0}, {6, 3, 0, 0, 0, 0, 0, 0, 0, 0},
{6, 2, 0, 0, 1, 0, 0, 0, 0, 0}, {6, 2, 1, 0, 0, 0, 0, 0, 0, 0},
{6, 1, 2, 0, 0, 0, 0, 0, 0, 0}, {6, 1, 0, 1, 0, 1, 0, 0, 0, 0},
{6, 1, 1, 0, 1, 0, 0, 0, 0, 0}, {6, 1, 1, 1, 0, 0, 0, 0, 0, 0},
{5, 4, 0, 0, 0, 0, 0, 0, 0, 0}, {5, 3, 0, 0, 0, 1, 0, 0, 0, 0},
{5, 3, 1, 0, 0, 0, 0, 0, 0, 0}, {5, 2, 0, 0, 2, 0, 0, 0, 0, 0},
{5, 2, 2, 0, 0, 0, 0, 0, 0, 0}, {5, 2, 0, 0, 1, 0, 1, 0, 0, 0},
{5, 2, 0, 0, 1, 1, 0, 0, 0, 0}, {5, 2, 1, 0, 0, 1, 0, 0, 0, 0},
{5, 2, 1, 0, 1, 0, 0, 0, 0, 0}, {5, 2, 1, 1, 0, 0, 0, 0, 0, 0},
{5, 1, 3, 0, 0, 0, 0, 0, 0, 0}, {5, 1, 2, 0, 0, 1, 0, 0, 0, 0},
{5, 1, 2, 1, 0, 0, 0, 0, 0, 0}, {5, 1, 1, 2, 0, 0, 0, 0, 0, 0},
{5, 1, 0, 1, 0, 1, 0, 1, 0, 0}, {5, 1, 1, 0, 1, 0, 1, 0, 0, 0},
{5, 1, 1, 0, 1, 1, 0, 0, 0, 0}, {5, 1, 1, 1, 0, 1, 0, 0, 0, 0},
{5, 1, 1, 1, 1, 0, 0, 0, 0, 0}, {4, 3, 2, 0, 0, 0, 0, 0, 0, 0},
{4, 3, 1, 0, 0, 0, 1, 0, 0, 0}, {4, 3, 1, 0, 1, 0, 0, 0, 0, 0},
{4, 3, 1, 1, 0, 0, 0, 0, 0, 0}, {4, 2, 3, 0, 0, 0, 0, 0, 0, 0},
{4, 2, 0, 0, 2, 0, 0, 1, 0, 0}, {4, 2, 2, 0, 0, 0, 1, 0, 0, 0},
{4, 2, 2, 0, 0, 1, 0, 0, 0, 0}, {4, 2, 2, 1, 0, 0, 0, 0, 0, 0},
{4, 2, 0, 0, 1, 2, 0, 0, 0, 0}, {4, 2, 1, 0, 0, 2, 0, 0, 0, 0},
{4, 2, 1, 2, 0, 0, 0, 0, 0, 0}, {4, 2, 0, 0, 1, 0, 1, 0, 1, 0},
{4, 2, 0, 0, 1, 1, 0, 1, 0, 0}, {4, 2, 0, 0, 1, 1, 1, 0, 0, 0},

```

```

{4, 2, 1, 0, 0, 1, 0, 1, 0, 0}, {4, 2, 1, 0, 0, 1, 1, 0, 0, 0},
{4, 2, 1, 0, 1, 0, 1, 0, 0, 0}, {4, 2, 1, 1, 0, 0, 1, 0, 0, 0},
{4, 2, 1, 1, 0, 1, 0, 0, 0, 0}, {4, 2, 1, 1, 1, 0, 0, 0, 0, 0},
{4, 1, 3, 1, 0, 0, 0, 0, 0, 0}, {4, 1, 2, 2, 0, 0, 0, 0, 0, 0},
{4, 1, 2, 0, 0, 1, 0, 1, 0, 0}, {4, 1, 2, 0, 0, 1, 1, 0, 0, 0},
{4, 1, 2, 1, 0, 0, 1, 0, 0, 0}, {4, 1, 2, 1, 0, 1, 0, 0, 0, 0},
{4, 1, 2, 1, 1, 0, 0, 0, 0, 0}, {4, 1, 1, 3, 0, 0, 0, 0, 0, 0},
{4, 1, 1, 2, 0, 0, 1, 0, 0, 0}, {4, 1, 1, 2, 1, 0, 0, 0, 0, 0},
{4, 1, 1, 1, 2, 0, 0, 0, 0, 0}, {4, 1, 1, 0, 1, 0, 1, 0, 1, 0},
{4, 1, 1, 0, 1, 1, 0, 1, 0, 0}, {4, 1, 1, 1, 0, 1, 0, 1, 0, 0},
{4, 1, 1, 1, 0, 1, 1, 0, 0, 0}, {4, 1, 1, 1, 1, 0, 1, 0, 0, 0},
{4, 1, 1, 1, 1, 1, 0, 0, 0, 0}, {3, 2, 2, 0, 0, 0, 0, 0, 0, 0},
{3, 2, 2, 2, 0, 0, 0, 0, 0, 0}, {3, 2, 2, 0, 0, 1, 1, 0, 0, 0},
{3, 2, 2, 1, 0, 0, 0, 1, 0, 0}, {3, 2, 2, 1, 0, 0, 1, 0, 0, 0},
{3, 2, 2, 1, 0, 1, 0, 0, 0, 0}, {3, 2, 2, 1, 1, 0, 0, 0, 0, 0},
{3, 2, 1, 2, 0, 0, 0, 1, 0, 0}, {3, 2, 1, 2, 0, 0, 1, 0, 0, 0},
{3, 2, 1, 2, 1, 0, 0, 0, 0, 0}, {3, 2, 1, 1, 2, 0, 0, 0, 0, 0},
{3, 2, 1, 0, 1, 0, 1, 0, 1, 0}, {3, 2, 1, 1, 0, 0, 1, 0, 1, 0},
{3, 2, 1, 1, 0, 0, 1, 1, 0, 0}, {3, 2, 1, 1, 0, 1, 0, 1, 0, 0},
{3, 2, 1, 1, 0, 1, 1, 0, 0, 0}, {3, 2, 1, 1, 1, 0, 0, 0, 0, 0},
{3, 2, 1, 1, 1, 0, 1, 0, 0, 0}, {3, 2, 1, 1, 1, 1, 0, 0, 0, 0},
{3, 1, 2, 1, 2, 0, 0, 0, 0, 0}, {3, 1, 2, 1, 0, 1, 0, 1, 0, 0},
{3, 1, 2, 1, 1, 0, 0, 1, 0, 0}, {3, 1, 2, 1, 1, 0, 1, 0, 0, 0},
{3, 1, 2, 1, 1, 1, 0, 0, 0, 0}, {3, 1, 1, 2, 1, 0, 0, 0, 0, 0},
{3, 1, 1, 2, 1, 0, 1, 0, 0, 0}, {3, 1, 1, 2, 1, 1, 0, 0, 0, 0},
{3, 1, 1, 1, 2, 1, 0, 0, 0, 0}, {3, 1, 1, 1, 1, 2, 0, 0, 0, 0},
{3, 1, 1, 0, 1, 1, 0, 1, 1, 0}, {3, 1, 1, 1, 0, 1, 1, 0, 1, 0},
{3, 1, 1, 1, 0, 1, 1, 1, 0, 0}, {3, 1, 1, 1, 1, 0, 1, 0, 1, 0},
{3, 1, 1, 1, 1, 0, 1, 1, 0, 0}, {3, 1, 1, 1, 1, 1, 0, 1, 0, 0},
{3, 1, 1, 1, 1, 1, 1, 0, 0, 0}, {2, 1, 1, 1, 1, 1, 1, 1, 0, 0}

```

-end of data-

APPENDIX D LABELED GRAPHS AND TREES

D.1 Number of Labeled Graphs and Labeled Trees

To obtain the number of labeled graphs with n vertices, we need only observe that each of the $\binom{n}{2}$ possible edges is either present or absent. Thus we have the following theorem:

Theorem D.1.1 [5] *The number of graphs on n labeled vertices is $2^{\binom{n}{2}}$.*

In Lovász [6], the Prüfer-code is used to denote a labeled tree. And, for any given Prüfer-code, one can construct the tree which is denoted by that code. Then, using the properties of the Prüfer-code, Cayley proved the following theorem:

Theorem D.1.2 (Cayley) [8] *The number of trees on n labeled vertices is n^{n-2} .*

D.2 Sketch of the Construction of a Prüfer-code

Among the end vertices of a given labeled tree, pick the one whose label is minimal. Cut it off from the tree and write down the label of its adjacent vertex. Repeat this process until only one edge remain. At this point we have a sequence of $n - 1$ numbers written down. We call this resulting sequence the *Prüfer-code* of the tree.

The following example shows how we construct the Prüfer-code of a given tree.

Example D.2.1 Let T be the tree shown below.

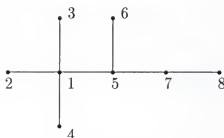


Figure D-1.

There are 5 end vertices, 2, 3, 4, 6, 8. We cut off the smallest labeled end vertex which is 2. Then the sequence is $\langle 1 \rangle$ and the remaining tree is Figure D-2 below.

Then we cut off the next smallest labeled end vertex which is 3, and add the label, 1, of the adjacent vertex of 3 to the sequence. That is, the sequence is $\langle 1, 1 \rangle$. From the remaining tree we cut off the next smallest labeled end vertex which is 4. Then the sequence is $\langle 1, 1, 1 \rangle$. And we get the tree shape in Figure D-3.

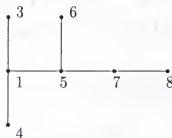


Figure D-2.

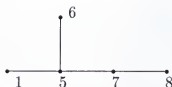


Figure D-3.

Among the 3 end vertices, 1 is the smallest labeled one. So we cut off the vertex 1 and write down the label, 5, of the adjacent vertex of 1. The sequence then will be $\langle 1, 1, 1, 5 \rangle$ and the remaining tree is Figure D-4.

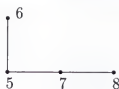


Figure D-4.

The remaining tree has two end vertices, that is, it is a linear shape. We cut off the smaller labeled vertex, 6. And the sequence will be $\langle 1, 1, 1, 5, 5 \rangle$. After we cut off the next smaller labeled vertex which is 5, and get the sequence $\langle 1, 1, 1, 5, 5, 7 \rangle$, the remaining tree has only two vertices. It's clear we have to cut off the smaller labeled vertex which is 7 and the adjacent vertex will be 8. Therefore we do not need to add this number to the sequence. Hence the last sequence, $\langle 1, 1, 1, 5, 5, 7 \rangle$, is the Prüfer-code of the given tree T.

If we assign different labels for the tree above, then we will get a different Prüfer-code. But the code must consist of 6 terms which is two less than the total number of vertices in the tree. And each term can be any number among $1, 2, \dots, 8$. Thus we have the following theorem.

D.3 Reconstruction of a Tree from a Prüfer-code

Let $\langle a_1, a_2, \dots, a_{n-2} \rangle$ be a Prüfer-code. That is, we expect an n -tree. Let $b_1, b_2, \dots, b_{n-1}$ be the indices of the removed points. Then, for any i , b_i is obviously different from $b_1, b_2, \dots, b_{i-1}, a_i$. Also, $b_i \neq a_j$ if $j > i$, because b_i is removed in the i -th step, so it can not be an adjacent vertex later.

Conversely, if $k \notin \{b_1, b_2, \dots, b_{i-1}, a_i, a_{i+1}, \dots, a_{n-2}\}$, then v_k is an end vertex of $V(T) - \{v_{b_1}, v_{b_2}, \dots, v_{b_{i-1}}\}$. Thus

$$b_i = \min\{k | k \notin \{b_1, b_2, \dots, b_{i-1}, a_i, a_{i+1}, \dots, a_{n-2}\}\}.$$

Let the sequence $\langle 1, 1, 1, 5, 5, 7 \rangle$ be a Prüfer-code. That is, $a_1 = 1, a_2 = 1, a_3 = 1, a_4 = 5, a_5 = 5, a_6 = 7$. Then $b_1 = \min\{k | k \notin \{a_1, a_2, \dots, a_6\}\} = \min\{k | k \notin \{1, 5, 7\}\} = \min\{2, 3, 4, 6, 8\} = 2$. That is, there is an edge between $a_1 = 1$ and $b_1 = 2$. Then $b_2 = \min\{k | k \notin \{b_1, a_2, \dots, a_6\}\} = \min\{k | k \notin \{2, 1, 5, 7\}\} = \min\{3, 4, 6, 8\} = 3$. Thus the tree T has an edge between $a_2 = 1$ and $b_2 = 3$. In a similar manner, we have the following:

$$\begin{aligned} b_3 &= \min\{k | k \notin \{b_1, b_2, a_3, \dots, a_6\}\} \\ &= \min\{k | k \notin \{2, 3, 1, 5, 7\}\} = \min\{4, 6, 8\} = 4, \\ &\quad (\text{Thus } T \text{ has an edge } \{a_3, b_3\} = \{1, 4\}) \\ b_4 &= \min\{k | k \notin \{b_1, b_2, b_3, a_4, a_5, a_6\}\} \\ &= \min\{k | k \notin \{2, 3, 4, 5, 7\}\} = \min\{1, 6, 8\} = 1, \\ &\quad (\text{Thus } T \text{ has an edge } \{a_4, b_4\} = \{5, 1\}) \\ b_5 &= \min\{k | k \notin \{b_1, b_2, b_3, b_4, a_5, a_6\}\} \\ &= \min\{k | k \notin \{2, 3, 4, 1, 5, 7\}\} = \min\{6, 8\} = 6, \\ &\quad (\text{Thus } T \text{ has an edge } \{a_5, b_5\} = \{5, 6\}) \\ b_6 &= \min\{k | k \notin \{b_1, b_2, b_3, b_4, b_5, a_6\}\} \\ &= \min\{k | k \notin \{2, 3, 4, 6, 7\}\} = \min\{5, 8\} = 5, \\ &\quad (\text{Thus } T \text{ has an edge } \{a_6, b_6\} = \{7, 5\}). \end{aligned}$$

And, for the vertex $a_6 = 7$, the largest labeled vertex v_8 will be the remaining adjacent vertex, that is, there is an edge $\{7, 8\}$.

So the tree T has the vertex set $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and the edge set $E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{5, 6\}, \{5, 7\}, \{7, 8\}\}$, which is the tree shape in the example above. That is, we can construct a labeled tree from a Prüfer-code. Clearly, different Prüfer-codes produce different labeled trees.

BIOGRAPHICAL SKETCH

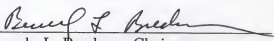
Tae-Il Yi was born December 16, 1962 in Seoul, Korea. He attended the Dankook University from 1981 to 1988, earning a Bachelor of Science degree in mathematics education in February 1988. During college, he was in the service of the Korean Army for 27 months. After college, he entered graduate school at Dankook University in March 1988, and completed a Master of Science degree in mathematics in February 1990. During that time he worked at Yongmoon High School as a mathematics teacher.

He married Moon-Sil Kim on June 27, 1987. Their first son, Han-Yong (David), was born April 12, 1988, and their second son, Chang-Yong (Peter), was born May 3, 1994.

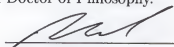
In August 1991 Tae-Il Yi entered graduate school in the Mathematics Department at the University of Illinois at Urbana/Champaign. After receiving the degree of Master of Science in August 1994, he moved to graduate school in the Instruction and Curriculum Department of the College of Education at the University of Florida, and in August 1995 transferred to the University of Florida, Mathematics Department. He pursued a concurrent degree in the College of Education and received a degree of Master of Education in December 1997. During his doctoral work in the Mathematics Department, he served as a graduate teaching assistant.

He received a Russell Corporation Endowed Dissertation Scholarship Spring 2000. He is a member of the Phi Kappa Phi honor society.

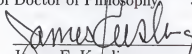
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Beverly L. Brechner, Chairman
Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Philip Boyland
Associate Professor of Mathematics

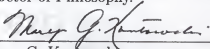
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


James E. Keesling
Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Li-Chien Shen
Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Mary G. Kantowski
Professor of Teaching and Learning

This dissertation was submitted to the Graduate Faculty of the Department of Mathematics in the College of Liberal Arts and Sciences and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 2000

Dean, Graduate School



.Y5

